2011

Predictive and adaptive game development a practical application of development models to the independent video game industry

Liam A. Hunt
Edith Cowan University

Edith Cowan University

Copyright Warning

Predictive and Adaptive Game Development: A practical application of development models to the independent video game industry.

Liam Hunt
Bachelor of Creative Industries

Faculty of Education and Arts

Submitted on the 11th of November, 2011

Abstract

Through the process of researching a number of game and system development models, this study defines two archetypical development models - the predictive and the adaptive. Using each of these models, two video games have been developed in order to measure the practical effect that they can have on the design, development process and player experience of a game. Using the knowledge gained through playtesting and an analysis of the relationship between each game and its development methodology, a number of development guidelines and recommendations have been specifically constructed for use by independent games developers. The practical development and written component of this project aim to answer the following research questions: how does choosing between a predictive and adaptive development model archetype affect the design, development process and player experience of a game?; and how can components from a variety of game design and development models be integrated into development guidelines and recommendations for independent video game developers?

The predictive development model archetype is primarily influenced by Royce's (1970) 'Waterfall' design model, while the adaptive development model archetype is primarily influenced by Keith's (2010) Agile approach. The way in which this study's final guidelines and recommendations are tailored to independent games development has largely been influenced by Kerr's (2006) analysis of the video game industry, as well as the experience and suggestions of a number of accomplished games developers. The entirety of this study has been conducted though a design-based research methodology within an action research framework. Through the analysis of this project's two games and their development processes, a dialog is created between different development methodologies (from both a games and system development background), as well as the theoretical and practical components of game development. Ultimately it is concluded that while predictive and adaptive development approaches to game development will affect a game in specific, predictable ways, independent developers must use a knowledge of development methodologies to select components specifically tailored to their individual projects.

I certify that this thesis does not, to the best of my knowledge and belief:

(i) Incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;

(ii) Contain any material previously published or written by another person, except where due reference is made in the text; or

(iii) Contain any defamatory material.

I also grant permission for the Library at Edith Cowan University to make duplicate copies of my thesis as required.

Signed, Date:

Liam Hunt

Contents

Introduction

According to Forbes' Mary Jane Irwin (2008), of all video games that enter into production, only 4% end up making a profit. Of the games that are actually completed and reach store shelves, still only 20% are profitable. Typically a game will have to be revised throughout its design process, increasing costs and development time, with "about 60% of a game's budget [being] spent reworking or redesigning a game" (Irwin, 2008). With such a small amount of video games being successfully completed and turning in a profit, it is important to have a formal understanding of game development processes, and the ways in which they can make or break a game before it even reaches a player's system.

Game design is more than planning out the story, gameplay features and assets of a game. The process involves designing these elements in such a way that they fit into a cohesive whole, allowing them to work together in bringing the player an experience that is memorable, enriching, and ultimately fun. In order to develop and structure the elements of a system effectively, it is important that we have a set of formalised models, processes and theories that can be employed to inform every aspect of its construction (Adler, 2005). This study involved investigating two identified game development methodology archetypes; predictive and adaptive, and practically analysing them in order to identify aspects of both that can be employed to effectively develop a game in an independent context.

With the advent of digital distribution services such as Xbox Live Arcade, Steam and the iPhone store, independent game development has become a much more viable industry with the shift of video games into a 'multi-medium' (Ip, 2008). The commercial and critical success of independent titles such as Minecraft, Super Meat Boy, World of Goo and Braid demonstrate that it is viable for independent developers to not only make a living though their craft, but to become internationally recognised as competitors to larger budget developers. With the growth of this sector of video games development, it is important to carry out an investigation into what formal game development methodologies and theories may be suitable for use in such an industry.

This study establishes of series of development recommendations and guidelines for use by small start-up teams or independent developers. These recommendations and guidelines have been synthesised though the combination of different elements from two separate

development methodology archetypes, namely predictive and adaptive development (both of which are explained in more greater later in this submission). The effects and implications of these two approaches have been identified through the thorough analysis of the development process and playtesting feedback of two games, one developed adaptively and the other developed predictively.

Research Questions

In order to practically investigate the ways in which the choice of a development model can affect the game it is used to develop, as well as devise formalised methods specifically designed to assist the growing industry of independent games development, two research questions were identified:

- How does choosing between a predictive and adaptive development model archetype affect the design, development process and player experience of a game?
- How can components from a variety of game design and development models be integrated into development guidelines and recommendations for independent video game developers?

The exploration of these questions has involved a definition of predictive and adaptive model archetypes, developed through an analysis of models and methodologies used in game development, traditional software engineering and project management. These development archetypes informed the development of two games (one using each model), which have been analysed to draw a clear comparison and contrast between the ways in which different development models and approaches can influence the design, development process and final player experience of a game. This analysis has been drawn from validated guidelines for effective design and explored within the context of a review of the role of independent games within the video game industry as a whole.

Literature Review

This literature review has been split into three main sections. The first section details various theories and studies concerning game design, as well as the ways in which games can be segmented into core components that can be analysed and measured. The second section explores independent games development, the various theories and practices that currently exist within this field, and its place within the modern video games industry. The final section investigates pre-established formal game and system development models, mainly concerning Royce's (1970) 'Waterfall' model and Keith's (2010) Agile game development with Scrum, including a comparison between predictive and adaptive archetypes.

Game design theory

In order to properly analyse the two games developed for this a project firm understanding of what a game is must be established, along with the fundamental components that make up gameplay and what comprises a good game. To measure the effect that predictive and adaptive development approaches can have on a game, a theoretical basis and game design lexicon must be determined, along with criteria that can be used to assess the quality of each developed game.

Huizinga (1955;2005) identifies the phenomenon of play as universal between cultures and species, used as a means to learn, relieve stress and have fun. He argues that play satisfies a human need beyond the purely biological, existing outside the realm of normal life, providing people with a simulated way to explore their limits and surroundings within an ordered and structured framework. Games are an example of such a framework used for the facilitation of play, structured by a set of rules or parameters and most commonly including an overall goal for the player to achieve. However, the specific components that must be included for a play activity to be considered a game are widely debated.

Juul (2005) provides a brief definition of a game as follows:

> A game is a rule-based system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels emotionally attached to the

> outcome, and the consequences of the activity are negotiable (Juul, 2005, p. 36).

Salen & Zimmerman (2004), in a simpler form, define a game as "a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome" (Salen & Zimmerman, 2004, p. 81). In both cases, rules are defined as an essential component of a game and necessary to the organised structure of play, whether they are implicit or explicit. Both definitions acknowledge that a game is a system which will result in an outcome that can be quantified; a player who reaches one of the outcomes of a game should be able to recognise and define it.

Oxland (2004) provides a practical analysis of a number of components that are inherent in all games, in terms of their function and application in video games.

- **Rules and boundaries** guide and structure gameplay, encapsulating the game space in order to remove player ambiguity in regards to what they are able or not able to do.
- **Feedback** is another method through which players can be guided, whether given explicitly or implicitly, exposing the game's rules and boundaries in various ways to promote proper gameplay.
- **Interface** consists of the way in which a player is allowed to interact (and thus play) with the game system, facilitating gameplay while virtually standing in for the movement of their own bodies in analogue games.
- **Goals, quests and challenges** are the quantification of a game's outcome (or many outcomes), providing the player with motivation for gameplay.
- **Balance** is the way in which all these components of a game are structured in order to complement each other and ultimately encourage the player to willingly play within the defined rules towards an outcome that they have some motivation to achieve.

Adams (2010) argues that the two conceptual components that create gameplay within a video game are core mechanics and user interface. Core mechanics are more specific than rules and are the implementation of a game's underlying mathematical systems and models. Core mechanics determine exactly what the player can do along with the subsequent effect that their actions will have on the game world. These mechanics can be expressed in an algorithmic form and drive the very heart of the game; without them, there would be no interaction. The user interface sits between these core mechanics and the player, translating

their inputs into perceivable outputs. This is parallel to Oxland's (2004) description of interface as an essential game component, both guiding and tutoring the player in what actions they may take, while acting as an embodiment of the rules and boundaries that limit and frame these actions.

The mechanics of a game are the underlying components that drive action and through their affordances and limitations, construct rules and goals. The game components identified by Oxland (2004) are all influenced by the mechanics that have been built into the game's fundamental systems. Jarvinen (2008) identifies a game's mechanics as the essential core of gameplay, working together to provide the player with access to the game in such a way that they are able to progress towards their goal. Individual mechanics are often simple, self contained tasks, such as aiming, shooting, moving and jumping, and in most cases are operated by the player in parallel (e.g. they may move and shoot simultaneously). Jarvinen (2008) classifies game mechanics into one of three categories: primary/sub mechanics are those that work towards the highest order goal for the player (e.g. jumping on an enemy in Mario). Modifier mechanics are those that change gameplay in such a way that the player changes their behaviours (e.g. collecting a star in Mario). Glocal mechanics are those that form the culmination of local goals into a global victory condition (e.g. reaching the end of the end of a level in Mario before the time runs out). In order for the mechanics and individual elements in a game to work in unison and construct a cohesive and playable whole, they must be balanced.

Balance can be described as a sense of equilibrium and harmony between challenge and achievement, mediated by feedback within the rules and boundaries of a game. Rolling & Adams (2003) define a balanced game as "one where the main determining factor for success of the player is the skill level of that player" (Rolling & Adams, 2003, p. 240). Commonly, balance is achieved within a game through rigorous testing, trials and tweaking, which can consume a large amount of time and effort on the part of the quality assurance testers. Rollings & Adams (2003) identify two broad classes of balance in video games: static balance and dynamic balance. Static balance involves constructing the rules and their interactions in such a manner that they are fair and avoid promoting dominant play strategies too greatly. Employing static balance aims to ensure that a game begins at a point of equilibrium, ensuring all players are afforded the same initial chance of success. Dynamic balance involves tweaking game elements during play in order to provide the player with

consistent challenge and avoid a sense of stagnation. A common example of such balancing is shifting a game's difficulty level to match a player's performance – if they constantly succeed the game gets harder, if they constantly fail the game gets easier.

For a game's elements to be balanced, the testers of that game should know how properly analyse the way in which they interact with the player and each other. Consalvo & Dutton (2006) propose a methodology for the qualitative study of games that is broken into four main areas: object inventory, interface study, interaction map and gameplay log. Object inventory is a catalogue of every object in a game including the ways in which the player can interact with it, the method and purpose of its use, the way it is represented, and the behaviours it promotes. Interface study observes the ways in which the player is able to interact with the game and the means through which the game communicates with the player, which allows for an analysis of which information is given privilege and the limits that are imposed on gameplay. Interaction map involves an examination of the choices players are given when interacting with NPCs (Non Playable Characters) or other players, documenting their limits, flexibility, meanings and significance to the development of the game as a whole. Gameplay log is an exploration of the game as a whole, specifically the gameplay that emerges from the combination of its individual elements. This final area is where a game's balance, or lack of balance, becomes apparent, following the ways in which the player is encouraged to act by the affordances and restrictions of the game mechanics and their subsequent enjoyment or frustration.

In order to illuminate bugs, balance issues, unusable mechanics or confusing game elements, many developers will playtest their games before final release. Playtesting generally involves gathering feedback from a number of potential consumers who have been allowed to test the game before it is released. Davis, Steury & Pagulayan (2005) propose a particular playtesting method whose goal is to "give game designers quantitative evaluations from players about how they experience the game" (Davis, Steury & Pagulayan, 2005). Their playtesting method involves one-on-one moderation in standardised environments, including large sample sizes of test subjects (approximately 25-35 people) who represent the entire demographic of the game's target audience. Subjects are given around an hour to play the game and construct their first impression and then are posed a number of questions that require standardized, quantifiable answers (e.g. a rating between 1 and 10). Testers subsequently interview the

subject in a more open ended manner, allowing them to justify their responses to the first set of questions and elaborate on their overall gameplay experience.

The playtesting method proposed by Davis, Steury & Pagulayan (2005) ultimately focuses on assessing a game's playability, problem areas and impact on a subject's overall experience, aiming to identify issues and problems encountered in order to find areas to improve. It and other playtesting methods are used to determine whether players will perceive a game as fun and enjoyable, challenging, innovative and novel. Davis, Steury & Pagulayan (2005) assert that questions posed to playtesting subjects should be asked in reference and comparison to other games they have played. By relating their experience playing a game to previous gameplay experiences, subjects can identify what worked and what didn't without having to be articulate in game design and mechanics.

An understanding of the elements of game design is crucial to both the task of developing two quality games and the construction of this study's guidelines and recommendations for independent game developers. These guidelines and recommendations are structured around addressing issues highlighted by the literature, such as the definition of rules and the balancing of mechanics, focusing on steps that directly influence the end experience of a final game. Therefore a study such as this benefits from the existing work done on playtesting and game analysis, in order to identify the strengths and weaknesses of specific games that can be used as a basis for exploring how different development approaches can contribute to the quality of the resultant games and their benefits for independent developers.

Significance of independent development in the modern video game industry

In order to adapt the above generic principles for use by independent game developers an understanding of the modern independent games industry must be established, including the way in which development within differs from that of the games industry at large.

According to a 2009 study by IpsosMediaCT (2010), 67% of U.S. households play computer or video games, with national video game sales reaching 10.5 billion dollars. Game players are 60% male and 40% female, with their average age being 34. The top selling genres of 2009 were sports, action, family entertainment and shooters, respectively. All of these statistics suggest that video games are assuming the mantle of a mainstream medium and the

average person who plays games is becoming increasingly closer to the statistically average person in society. The increasing diversity of the audience that games can be targeted towards suggests an opportunity for increasingly diverse gameplay experiences, with the rise of smart phones and the internet allowing for low budget games to reach and appeal to a global market.

As asserted by Ip (2008), the increasingly diverse video game audience can be attributed to the convergence of new technologies and content with the games industry. The integration of technologies such as social networking, motion controls and downloadable content has resulted in the establishment of new genres and forms of gameplay, targeted at demographics that may have previously been considered 'non-gamers'. Ip (2008) demonstrated that video games have become a 'multi-medium', consisting of elements traditionally associated with other media such as books, film and music. He asserts that the video games industry is developing into "a market in which emphasis is firmly placed on users, user-generated content, and the idea of universal access" (Ip, 2006, p. 220).

Kerr (2006) identifies the games industry as a 'cultural industry', noting its general high risk and production costs, low reproduction costs and low ratio of product financial success. Independent, small or 'mini' games are characterised as possessing a number of revenue models and involving "shorter development cycles and lower production costs" (Kerr, 2006, p. 61). Such games thus allow more experimental and less proven concepts, due to their lower risks.

Kerr (2006) observes that the majority of games are developed or owned by those that fund them (the publisher), who thus are justified in overseeing management. This results in restricting of creative control and aggressive negotiations with contracted developers, due to the fact that the publisher is the one that bears the initial financial risk and investment. He argues that the current structure of the video games industry is focused on maximising economic value, subsequently seeking to minimise risk and focus on proven and marketable concepts. Although Kerr (2006) condemns this as stifling innovation and creativity, he rationalises it by pointing out that a company will always be primarily focused on profit and asserts that independent developers still exist that, alongside with academics, artists and user groups, still "contribute to overall innovation and diversity in the industry" (Kerr, 2006, p. 73). Due to the high cost, high risk nature of the modern video game industry, low budget

independent games development has become a testing ground for new ideas, and an area where new developers can develop and market their games without the necessary publisher relationship inherent in the larger budget areas. Economically, this has become sustainable through the rise of digital distribution markets such as Xbox Live Arcade, Steam and the iPhone market.

Although independent games generally have a significantly smaller user base than large budget mainstream games, they still have the potential for large profit margins due to their relatively lower budgets. Shakar & Bayus (2002) compare the unit and software sales between two systems made by Sega (higher user base) and Nintendo (lower user base), from 1993 through to 1995. Shakar & Bayus (2002) argue through their analysis and subsequent findings that Nintendo was able to gain greater profit through a stronger network of users, and that a large customer network does not necessarily guarantee high sales. Rather than designing a video game or system to target the largest potential audience, it can be better to specifically target a smaller group of people and use less development resources. By building confidence in a niche user base, independent developers can construct the foundations necessary to develop games using a progressively larger budget, attracting a progressively larger audience through the strength of their base network.

In order to successfully develop independent games, game developer Jacob Stevens (2009) provides a number of tips specifically designed to be used in low budget and small team ventures. He argues that in order for an independent team to be successful all members must have demonstrated independent self-motivation and the team must choose the correct distribution platform for their design idea. Other preliminary methods that can be put in place before development of the game begins include planning the game around the team's pre-established strengths (not attempting to learn a large number of new skills) and setting limits on gameplay and scope in order to refine and limit the concepts that are created. As an approach to the practical development of the game, Stevens (2009) argues that iterative prototyping and play testing is the best way to ensure the final game is successful. He emphasizes the importance of involving test players and their constant feedback in the development process. Contrary to systems development methodologies, Stevens (2009) plays down the importance of design documentation, arguing that in a small team time would be better spent on the development of the game itself.

In an interview published by Christian Nutt (2011) of Gamasutra with a number of successful independent game developers, Jake Kazdal (developer of the video game *Rez*) echoes Steven's (2009) insistence of including outsider parties to playtest the game at different stages, in order to ensure that it can be understood by those who did not develop it. Kazdal attributes the success of his games to merciless marketing and demonstration to the people who would be potential buyers, highlighting the folly of independent developers solely focusing on the game's practical development. In the same interview by Christian Nutt (2011), Daniel Cook of independent development company Spry Fox agrees that prototyping is necessary to ensure a game remains fun and playable, in addition to iterating as often as possible. Cook insists that all members of an independent development team must work well together and should respect the act of designing a game as completely separate to other disciplines such as art or programming.

Although acknowledging the independent games industry as a source of new ideas and innovation, Grossman (2003) asserts that independent developers would also benefit from the ideas and lessons learnt by those in the mainstream games industry. In a series of post-mortems by experienced developers, he examines the different things that went right and wrong in a number of commercial game projects. Most developers cited a lack of planning or foresight into game design choices as being as source of trouble, often extending development times and costs, as well as forcing the development team to cut features from the project. In many cases, a lack of management or proper deference to a central creative source created inconsistencies and misconceptions during the development process. This caused the final game to be incoherent and lacking unity, or forced necessary revisions and re-design. Grossman (2003) notes that, for start-up independent developers, ambition must be restrained and a project's limits must be defined early in order to avoid the feature creep and overtasking inherent when developing with youthful, enthusiastic teams. In contrast to accomplished mainstream developers, he labels the inexperience of most independent developers as both beneficial and perilous. Such developers are more likely to try new methods, ideas and innovations in development due to their lack of engrained knowledge but can often ignore vital practical elements of developing a real, playable game in favour of being caught up in the romanticism and excitement of their dream concept.

In order to construct effective game development guidelines and recommendations for independent developers the recommendations of successful developers in the field must be

taken into account, while considering the reflections and post-mortem advice of professionals working with a larger budget. As the independent games market has only relatively recently become an economically viable means of professional games development, the unique nature of its place in the industry at large will play a large role in the final recommendations for new developers. In this sense, areas that require consideration include the implications and practical impact of working alone or in a small team, developing on a small budget with a low level of experience, releasing content in a digital market, and the ways in which these factors will affect the structure of a game development process.

Formal game and system development models

In order to apply game design theory and an awareness of the independent games industry to the development of a successful game, one must have a formal, proven plan or model to structure it around. Valuable guidelines and recommendations for independent games development can only be constructed through an understanding of existing games development models, as well as the systems development models they were adapted from and influenced by.

There are a large number of pre-conceived approaches to traditional systems and software development, described by the USA Department of Health & Human Services as a "framework that is used to structure, plan, and control the process of developing an information system" (Department of Health & Human Services - USA, 2008). Some of these methodologies include Waterfall, Prototyping, Incremental and Spiral. Each of these methodologies are structured in a linear or iterative fashion, or sometimes a combination of both. Adler (2005) argues that such formalised and structured models when developing a system are necessary in order to ensure that it is completed within a reasonable timeframe and budget, stressing the fact that within professional fields only a small percentage of projects are adequately finished within the initial time and cost allocations. He asserts that having a planned approach and project schedule before development begins can minimise the already high risks within the chaotic field of software development. Specifically, Adler (2005) identifies a number of ways in which development models can assist companies including bureaucratic rationalisation in order to reduce uncertainty, modularity in order to reduce interdependence, and automation in order to reduce task complexity.

The necessity and benefit of using a formal model in order to structure the development of large software systems is further explored by Royce (1970). He argues that although all software systems must go through two common steps of analysis and coding, any large project will be doomed to failure unless proper time and effort is also allocated to the identification of the system's requirements, documentation, planning, designing and testing. It is here that Royce's (1970) model of software development is introduced, later dubbed by others as the 'Waterfall' approach. In a basic sense, the model consists of 7 steps: system requirements, software requirements, analysis, program design, coding, testing and operations. In order to ensure that the final project is as user friendly, bug free and coherent as possible, Royce (1970) suggests that once initial requirements have been identified, approximately one third of the development time should be allocated to creating an initial prototype of the final system that is complete in intended functionality. This practice is intended to reduce the resources used in the project's test phase, allowing for the developers to practically test and revise their system long before it reaches a customer's hands.

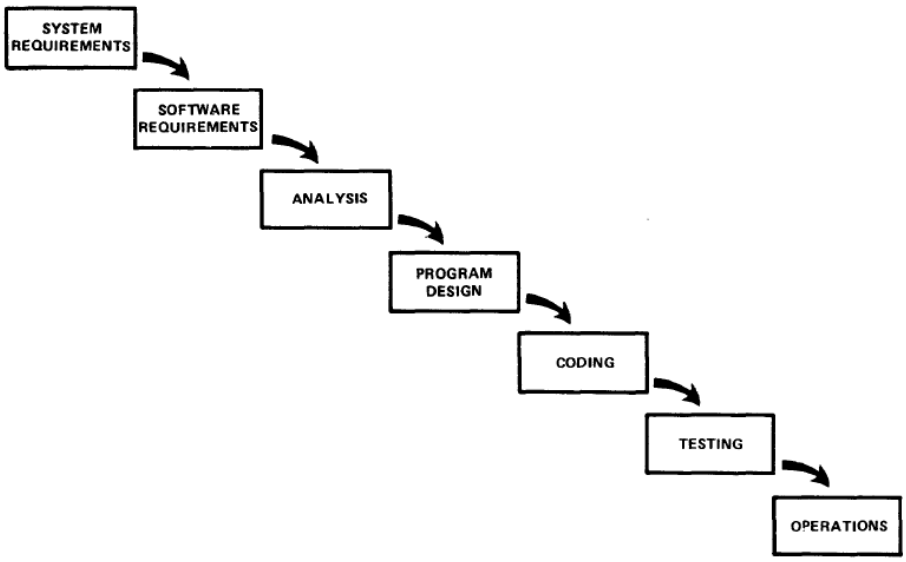

Figure 1. Waterfall model diagram. (Royce, 1970, p. 329).

Overall, Royce's (1970) model is structured in a mostly linear manner, but involves an iterative aspect in the preliminary prototyping of the system. It could be described as predictive as it involves an identification and definition of the final system and its functionalities prior to its actual development or coding phase. The USA Department of Health & Human Services (2008) describes its strengths as being measurable in terms of

costs and time, and allowing new team members to enter the design process while using the established documentation to understand the project's overall scope. Its weaknesses are cited as mainly involving its inflexibility and lack of room to respond to any changes or alterations, especially towards the end of the project.

In the field of traditional software system design, the USA Department of Health & Human Services (2008) identifies a number of other commonly used formal models, each of which can be used to suit specific types of projects depending on their strengths and weaknesses. Prototyping involves breaking a project into small manageable segments, involving the user or client throughout the entire process. Its strengths include greater user satisfaction, flexibility and quick identification of problems. However, its weaknesses include added costs and time, lack of documentation and potentially false user expectations. Incremental is described as both linear and iterative, involving either a series of iterations of Royce's (1970) 7 steps, or the use of his model's first 4 steps to plan and document, then developing and coding through Prototyping. Its strengths are described as allowing exploitation of knowledge gained from incrementing and allowing formal control when planning the project. Its weaknesses include a lack of consideration for business and technical requirements as well as encouraging difficult problems to be put off to demonstrate early success.

Another software development model that combines both linear and iterative frameworks is the Spiral model. Proposed by Bohem (1986), Spiral development involves breaking down the completion of a project into a number of cycles, all of which follow a common set of increasingly larger steps. Each cycle begins with a determination of the objectives, alternatives and constraints of the current aspect of the project, followed by an analysis and resolution of the potential risks involved. At this stage a prototype of the current project is designed and developed, then verified and prepared for use in the next development cycle. The final step of each cycle involves planning for the next cycle in the project, gradually increasing in scope as the development process progresses.

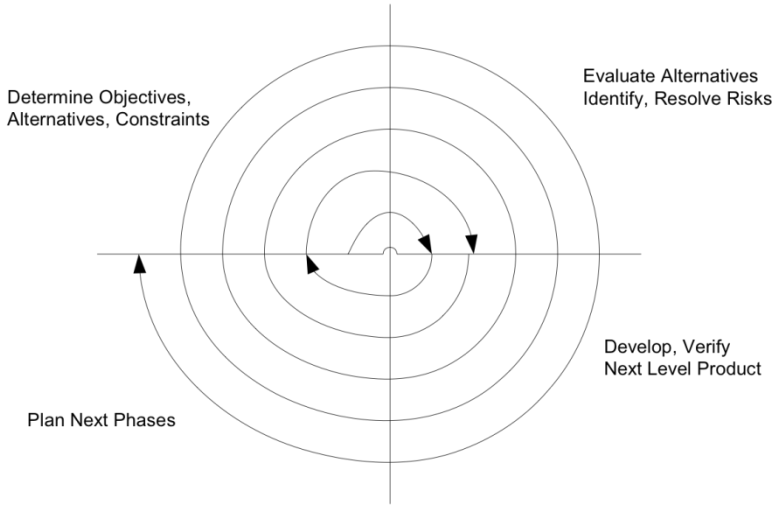

Figure 2. Spiral model diagram. (Department of Health & Human Services – USA, 2008, p. 6).

The USA Department of Health & Human Services (2008) identifies the Spiral model's focus on risk analysis and avoidance as its major strength. Additionally it highlights the fluidity of each software cycle's methodology, allowing for the adoption of Waterfall, Prototyping or Incremental approaches in each iteration, depending on the project's current needs. Conversely the Department of Health & Human Services (2008) cite this fluidity as a potential weakness due to the fact that there are no definite or exact steps to follow, thus requiring the assistance of a skilled project manager. Additional weaknesses include the lack of firm deadlines and Spiral's relative complexity in comparison to other development models.

Hayes & Games (2008) discuss ways in which the use of systematic and formal development processes can educate on the subject of game design itself. They argue that it is important for game design novices to structure their work around a pre-conceived pattern, taking time to plan and consider the design and overall structure of their game (both on a macro and micro scale) before undertaking the task of actually coding and building the game itself. Hayes & Games (2008) stress the need for enhanced focus on teaching better design thinking to novices, including "more sophisticated and useful understandings and practices" (Hayes & Games, 2008, p. 328). Through the undertaking of developing a game though a proper educational scaffolding, game design practices can be taught and practically built upon by 'learning by doing'. Formal development models can facilitate learning by providing an

evenly rounded development process, such as by allowing time for analysis, planning, development (coding) and testing. They can be used to guide the uninitiated so that they do not jump straight into coding and miss out on the overarching lessons that the other stages can teach about game design on a more macro scale.

The purpose of using a formalised development model that allows for iteration and re-adjustment is recognised in the Agile methodology of game development. Keith (2010) discusses the use and application of this methodology through an iterative development framework named Scrum, specifically tailored to break down the process of creating a game into a series of task focused 'sprints'. In order to facilitate these sprints, a game is broken up into groups of related tasks or features that must be completed and are documented in a project backlog. Every two to four weeks the game development team will meet to re-assess the current state of the game, selecting a number of tasks from the backlog that they can independently work on to fashion into a playable, potentially shippable release of that specific feature. Keith (2010) argues that as much new knowledge is generated during the design process, constant iterations are needed in order to incorporate this knowledge into the final product. He stresses that working prototypes, playable demonstrations and constant adaption are integral to producing a fun, coherent game, rather than comprehensive design documents and pre-planning.

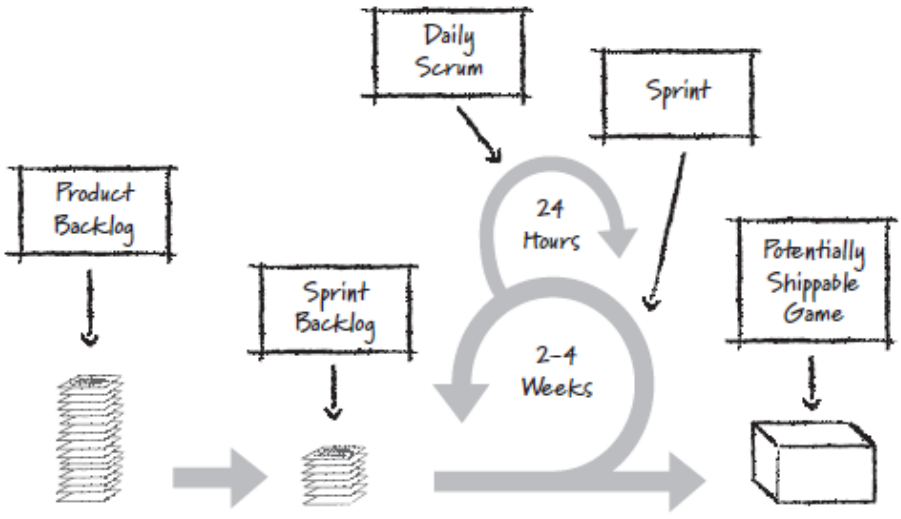

Figure 3. Agile model diagram. (Keith, 2010, p. 39).

Aside from being iterative by being structured around a series of task iterations and revisions, Keith's (2010) Agile game development model with Scrum could be labelled as adaptive. In this sense, although a final game is planned from the beginning of the development cycle, the developers constantly adapt the game's features (and thus final form) throughout the development cycle, in response to feedback from prototyping and demonstrations. This differs from other more archetypically predictive development models such as Royce's (1970) model by allowing a compromise of the initial project vision. Such a model identifies the development process as an evolving entity that the developers must work to accommodate, as opposed to something that can strictly be controlled by tight management and comprehensive documentation.

Bohem's (1986) Spiral model is an example of a model which can potentially sit between the adaptive and predictive archetypes. It seeks to predict the generic process that every iteration of a project's development can undergo, while allowing for multiple and increasingly complex development cycles in order to adapt and inform the final product. Depending on the current needs, identified risks and consumer demands at any stage in the development process, the spiral model can waver between the two archetypes, allowing for the introduction of elements from other development models.

Overall, predictive models would be preferable when there is a pre-defined customer expectation or specific structure the game must withhold at all costs, allowing for a definite vision of the final product to be established long before it takes a playable form. Adaptive models encourage change and thus will not usually allow for all aspects of a game to be planned in unison, seeking to allow a game's final project to be a direct response to its development process and the lessons learnt within. Through the analysis of different software development models however, one can see that such models are rarely purely adaptive or predictive, often incorporating elements primarily from one archetype but possessing a few from the other.

Conclusion

Although many theories have been established in the field of games culture and game studies, the technical development and methodologies of game development have been addressed in less depth. The majority of existing development models have been created for use in the

field of software systems development and, while they can be adapted for game development, are not specifically tailored for use by games developers. Due to its roots in systems development and programming, many elements of game design are often assumed or accepted and are rarely criticised, deconstructed or analysed for use in specific scenarios. As shown by the section on formal game and system development models, there are relatively few examples of a formalised approach to structuring the design of a game within a pre-conceived and tested model. The majority of methodologies taken and used by developers can be described as predictive; comprehensively planning as a separate task prior to actual development, or adaptive; using multiple iterations and prototypes to shape a game and its design based on feedback and analysis.

Overall, there has been an extremely small amount of formal investigation into the independent games industry and the different methods in which games can be developed in an independent context. The majority of literature on the subject is largely anecdotal, or is constructed from interviews and discussions with developers concerning their personal experience. However, it has been demonstrated that the popularity and economic value of independent games is growing through different channels and methods than previously used in mainstream games, in some cases even outselling and outperforming their higher budget brethren. In light of this growth, this study endeavours to fill a portion of the apparent theoretical gap, incorporating various existing theories with the practical design, development and analysis of two games. The two games have been developed with an understanding of effective game design theories, and have been analysed in terms of their rules and boundaries, feedback, interface, goals, challenges and balance, using a series of formalised testing methods. The findings resulting from the creation and analysis of these games will be used to construct a series of guidelines and recommendations to guide inexperienced independent game developers.

Theoretical Framework

In order reach a common understanding of the salient characteristics of predictive and adaptive development archetypes for the purposes of this study, their differences must be discussed in order to avoid any later confusion.

Predictive and adaptive development models can be distinguished by their approach to the planning and structure of a project. A predictive model will follow a top-down approach, specifying and planning a final project in its entirety before it enters development. In this sense, the use of such a model will generally be executed in a linear fashion, completing each aspect of the project before moving on to the next, with the entire development process being structured around documentation devised in the initial planning stages. An adaptive model will follow a bottom-up approach, involving minimal initial planning and documentation, instead initially focusing on development. Aspects of the project are developed iteratively, placing a large focus on analysis, revisal and re-designing.

Overall, predictive models would be preferable when there is a pre-defined customer expectation or specific structure the game must withhold at all costs, allowing for a definite vision of the final product to be established long before it takes a playable form. Adaptive models encourage change and thus will not usually allow for all aspects of a game to be planned in unison, seeking to allow a game's final project to be a direct response to its development process and the lessons learnt within.

Figure 4 demonstrates a sample framework for a predictive development archetype while Figure 5 demonstrates a sample framework for an adaptive development archetype.

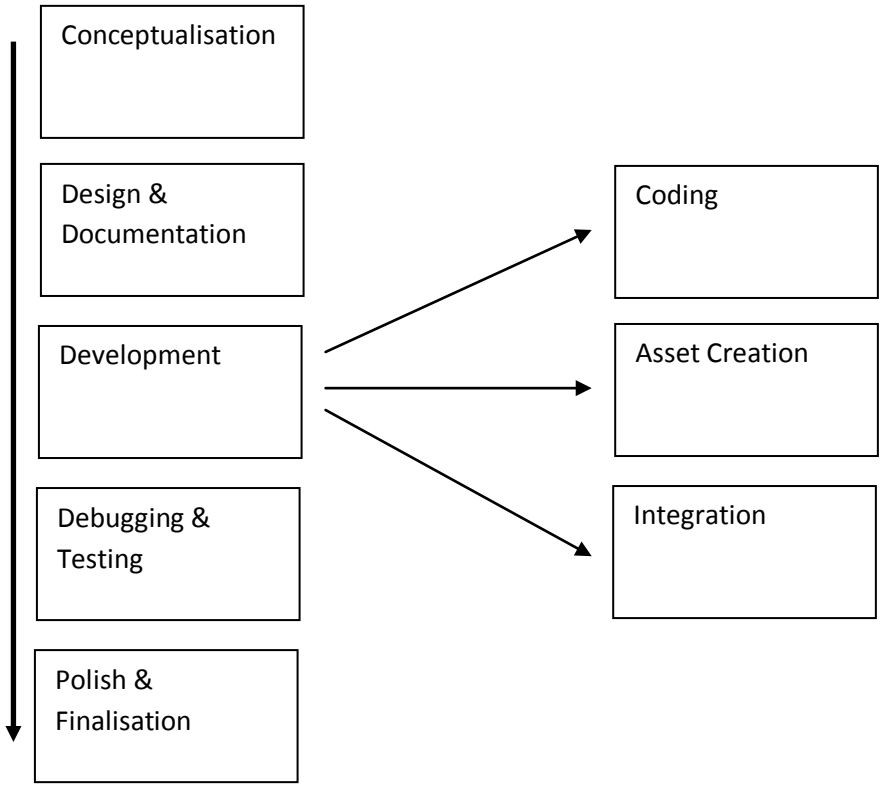

Figure 4. Predictive development framework

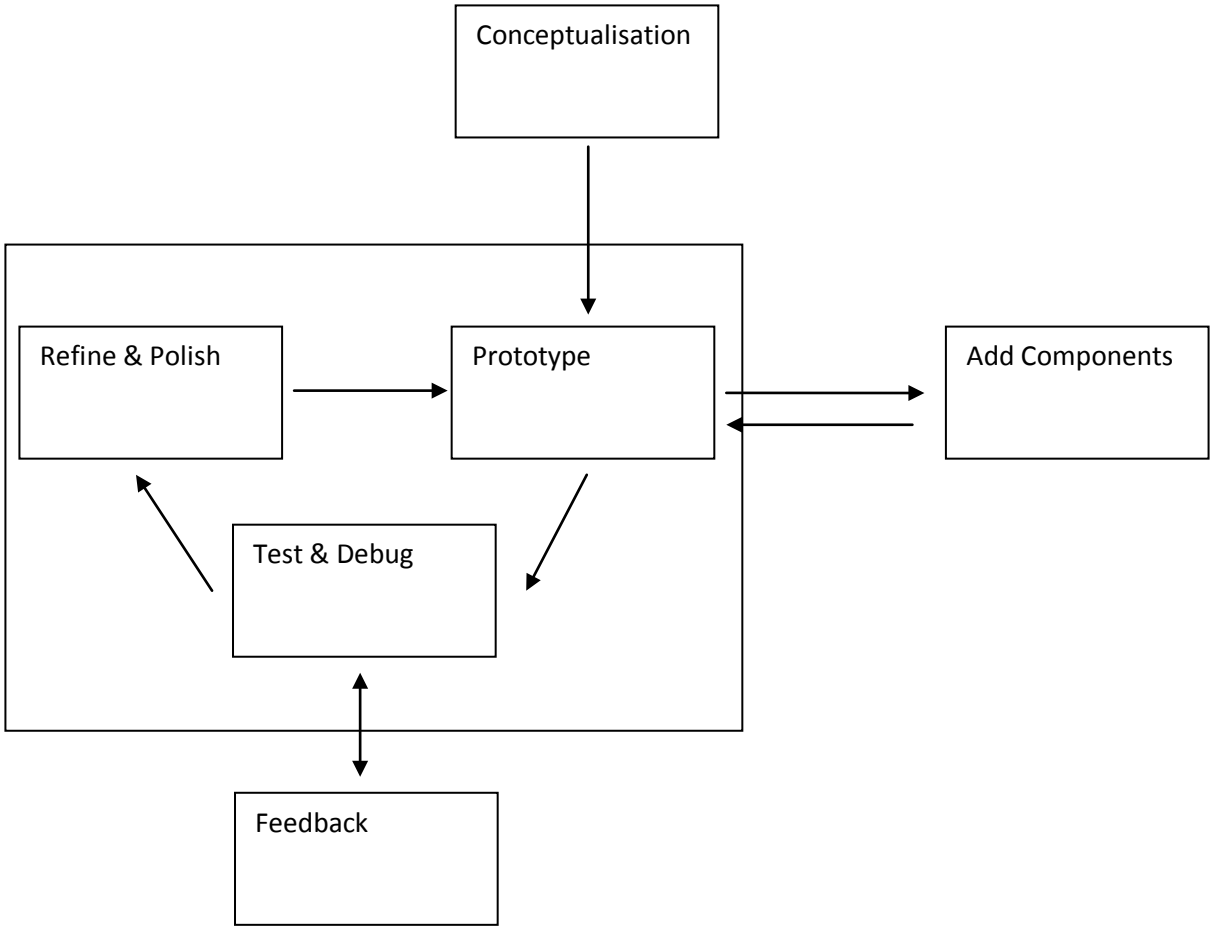

Figure 5. Adaptive development framework.

In many cases, the different salient characteristics of predictive and adaptive development methodologies can be expressed as opposites. Ideally, the type of hybrid development methodology approach that has been recommended for use by independent game developers would likely possess a mix of characteristics that would sit somewhere between those of a predictive or adaptive approach. Table 1 represents these competing characteristics as a continuum, where approaches may sit at a particular point on each of the dimensions identified.

Table 1. Predictive – adaptive characteristic continuum.

Predictive		**Adaptive**
Linear		Iterative
Focused on documentation		Minimal documentation
A broad definition of the game early in development		Game features are developed, then later synthesised
Restriction of changes to the initial concept		Refinement and adaption of the initial concept
Testing and debugging discrete from content development		Integration of testing and debugging throughout development process
Sequential creation of final game components from scratch		Game components are prototypes, then iteratively built upon and improved

Methodology

This project involved an investigation of archetypical theoretical development models for the development of video games, including the construction of two games based upon these models. For the purposes of the project, the term archetype has been defined as "the original pattern or model from which all things of the same kind are copied or on which they are based; a model or first form" (Dictionary.com Unabridged, n.d.). Criteria identified in the literature review was used to gauge the quality of the elements in each game and identify any problem areas or outstanding issues. The differences in design, development process and player experience of the two games (from both the researcher's and playtesters' perspectives) was then triangulated in order to first identify the contributing effect of their different development methodologies, and then inform the construction of a set of guidelines and recommendations specifically tailored for independent games development. As such, the methodology can be defined as design-based research within an action research framework.

Action research is defined by O'Brien (1998) as "learning by doing", involving a researcher identifying a problem, reflexively endeavouring to solve it, measuring success and potentially embarking on another method of solution. Overall it is structured to shape practical work through theory, using the results of that work to transform the existing theories. The entire research process and methodology of this project will be structured around the core focus of action research; completing a task in order to improve the theories that informed it.

As detailed by Baumgartner et al (2002) design-based research involves undertaking of research through theory driven design, enactment (implementation), analysis and re-design of a system. As a sub-set of action research, design-based research involves the combination of theory and practice into an iterative process, specifically to facilitate the reflexive development of a system that will be analysed in order to build upon the theories that informed its construction. Ultimately, design-based research attempts to combine informed theoretical research with practice, and thus is suitable in such a practical field as game design.

The basis of this project's methodology in action research facilitates the completion of a task multiple times (practically testing an archetypical game development model) in order to use the lessons learnt from those iterations to improve the quality of a final product (the guidelines and recommendations for independent games development). This adheres to the

methodology of action research through the task of developing two games independently and measuring their success in order to address issues and improve development theories in independent games development. Through an analysis of the two games that have been developed, the final outcome (strengths, weaknesses, and structure) of each game has been linked to the development approach used to create it. In this sense, the use of design-based research has assisted the project through its ability to "generate plausible causal accounts because of its focus on linking processes to outcomes in particular settings" (Baumgartner et al, 2002, p. 6). Through the exploration of the cause/effect relationship between the final product of a game and the development methodology used to construct it, guidelines and recommendations will be constructed for independent games development in an endeavour to capitalise on the strengths of each, while avoiding their weaknesses.

The merging of theory and practice makes design-based research within an action research framework an ideal methodology for this study. As evidenced in the literature review, there exists a segregation between ideologies and development methods in the video games industry. In an independent context, a final game product would benefit from increased developer awareness of the effects and implications of the identified archetypical predictive and adaptive approaches. Through the use of the previously detailed research methodology, both development approaches have been allowed to inform the final guidelines and recommendations for independent games development at the conclusion of this study.

The method through which this study endeavoured to answer the research questions can be broken down into three individual tasks for each question. To recap, the first question is:

- How does choosing between a predictive and adaptive development model archetype affect the design, development process and player experience of a game?

To investigate how development models affected the design of a game, a retrospective study has been conducted upon the two games created, using the qualitative method outlined by Consalvo & Duttton (2006). This study includes an object inventory, interface study, interaction map and gameplay log, looking at the meanings and promoted player experiences within the games. Through keeping a development journal throughout the process of creating both games, an analysis has been undertaken of the problems, issues, successes and failures that materialised as a result of their respective development methodologies and thus the ways in which they affect the games' development.

To investigate how different development models can affect a player's experience of a game, a series of playtests were constructed using the method outlined by Davis, Steury & Pagulayan (2005). During these playtests, the researcher sat with select subjects while they played each game, observing the way in which they behave and react to the game elements. Each playtesting participant, after playing each game, was posed with a set of standardised questions asking them to numerically rate certain aspects of the game and respond to a set of statements on the Likert scale (strongly agree to strongly disagree). Following these quantitative questions the subject was asked three open-ended questions, in order to identify any areas they specifically enjoyed, areas they deemed problematic, and any possible changes or additions to the game they would recommend.

The second research question is:

- How can components from a variety of game design and development models be integrated into an effective process for independent video game developers?

A variety of game design and development models have be identified through research (featured in the literature review), and used to construct a definition of two methodological archetypes: the predictive and the adaptive. The two games subsequently created for this project were then each developed according to one of these archetypes. To identify effective processes and practices for independent video game developers, the unique methods used by a number of successful developers within the independent field were researched and investigated. These processes were framed within the current state of the video game industry at large, as well its smaller independent sector.

As per the first research question, the two games created have been analysed in terms of the ways in which their development archetype affected their design, development and final player experience. This analysis was applied to the findings on independent video game development and the independent video game industry, in order to construct guidelines and recommendations for independent games development. These guidelines and recommendations were tailored towards the needs of independent developers and the demands of both the modern video game industry and modern players, in order to facilitate a unique process useable by new and inexperienced developers.

Table 2 summarises the approach taken to address the research questions. For each question a range of specific outcomes has been identified with the data required to explore those

outcomes and the analytical approach taken with this data. Triangulation has been possible through the use of multiple forms of data and analytical approaches that enable comparison between these.

Table 2. Summary of research outcomes, analysed data and analysis approach.

Research Outcome	Analysed Data	Analysis Approach
Development model effect on design	• Final games • Design documentation	Consalvo & Dutton's (2006) qualitative method
Development model effect on development process	• Development journals	Constant comparative analysis
Development model effect on player experience	• Playtesting feedback • Playtesting observations	Statistical and textual comparison
Identification of effective processes and practices for independent video game developers	• Literature review; development models and independent development	Textual content analysis
Construction of guidelines and recommendations for independent games development	• Results of the previous research outcomes	Triangulation

Limitations

A number of factors limit the usefulness of a direct statistical comparison of the playtesting data gathered for this project. Due to the iterative nature of this project it can be reasonably assumed that the development of the first game, including the skills gained and lessons learned by the developer, influenced the development of the second game and thus resulted in higher playtesting scores. Additionally, although both games were developed to be of a similar genre they contain different mechanics and core challenge types, which in turn could have influenced playtester feedback based on the gameplay styles they generally prefer.

In order to combat these limitations, the discussion of the way in which each development methodology archetype affected the design, development process and player experience of its respective game focuses on identifying recurring instances where salient aspects of the

development approach affected definable elements of each game. These effects on game elements do not include those consciously and deliberately added by the developer to adhere to the game concept, rather focusing on highlighting unforseen areas of differencing balance, polish, playability and prioritisation.

Development

The development of the two games took place over the period of 10 weeks, allowing 5 weeks for each. The first, titled 'Dematerialized', was developed according to an adaptive archetype while the second, titled 'Cursed', was developed according to a predictive archetype. During the development of each game a daily development log was maintained, detailing the tasks that were completed every day, the amount of time spent on each task and any problems or issues that arose during development. In addition to these logs, at the end of every week of development a weekly of build of each game in their current state was saved, in order to be able to practically analyse the way in which they evolved from a simple concept to a fully playable game. A consolidated summary of the development process of each game is detailed below, including graphical representations (Figures 10 & 11) of the number of hours spent on each generic task during the games' development (derived from the development logs).

Dematerialized

As the first game was developed according to an adaptive methodology, no formal planning was undertaken before development began. The game was based around a loose pre-planned concept, driven by providing the player with the ability to freely teleport around a 2D platforming environment. The initial concept was planned to be primarily involve combat (psychomotor) challenges with some minor puzzle elements, focusing on using teleportation to outsmart enemies and quickly attack them from multiple directions.

Week 1

The first week of development primarily focused on the construction of the underlying player driven mechanics, as well as an animated and controllable player model. The player model's appearance was designed based on two factors; making it fit in with a 'virtual' visual motif, and allowing it to be animated quickly and easily. Animations were created based on what looked good within the constraints of the model. The player attacking system was originally designed to be a three hit combo (later revised to two) and was planned to be further developed for use in the game's original beat-em-up concept.

The biggest initial difficulty that was encountered during development was the teleportation mechanic, largely due to the fact that 2D mouse movement needed to be interpreted in a non-orthographic 3D scene (causing issues with raycasting calculations). Additionally, systems

needed to be put in place to stop the player from being allowed to teleport into solid objects, which could not be detected (to the developers knowledge) with the raycast functions build into Unity. Because of these mechanics based difficulties, the majority of time spend during week 1 was spent on programming tasks, resulting in a very bland look in the weekly build (shown in Figure 6), but setting up the core gameplay for the final game.

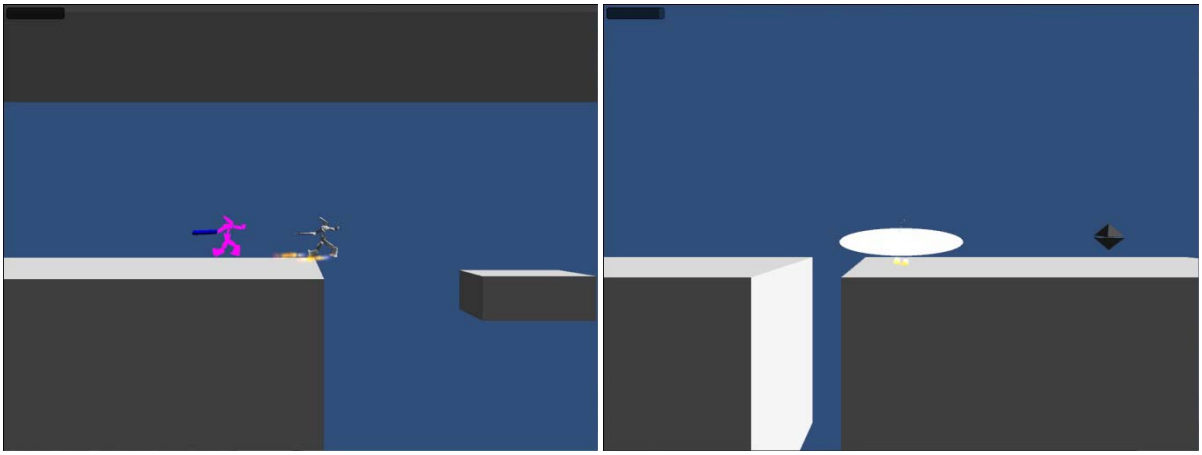

Figure 6. Dematerialized week 1 build.

Week 2

Week two was characterised by the start of proper level planning and development, and was the point in the game's development process where its overall concept shifted towards more of a focus on puzzle elements rather than combat challenges. This shift was primarily due to the realisation of the amount of time it would take to introduce good melee fighting mechanics, from both a programming and animation perspective. Instead, a decision was made to have the main challenges of the game stem from the interaction of the main teleportation mechanics with a combination of simple and predictable entities (teleport jammers, basic enemies, level geometry). Dematerialized's adaptive development approach helped it in this way by allowing it to be adapted to a revised scope, but also hindered it in the compromise of its initial concept and the redundancy of previously constructed elements that never made it into the final game (e.g. the 3rd attack animation, the 'intelligent' player clone enemy).

No insurmountable problems were encountered in week two, allowing both the creation of number of new elements and some time to be spent on the visual and aesthetic design of the game, as demonstrated in Figure 7. The general framing story was devised and introduced in

the beginnings of a basic tutorial level, but in many ways was tacked on as it was not originally planned to be a part of the game from the beginning of development.

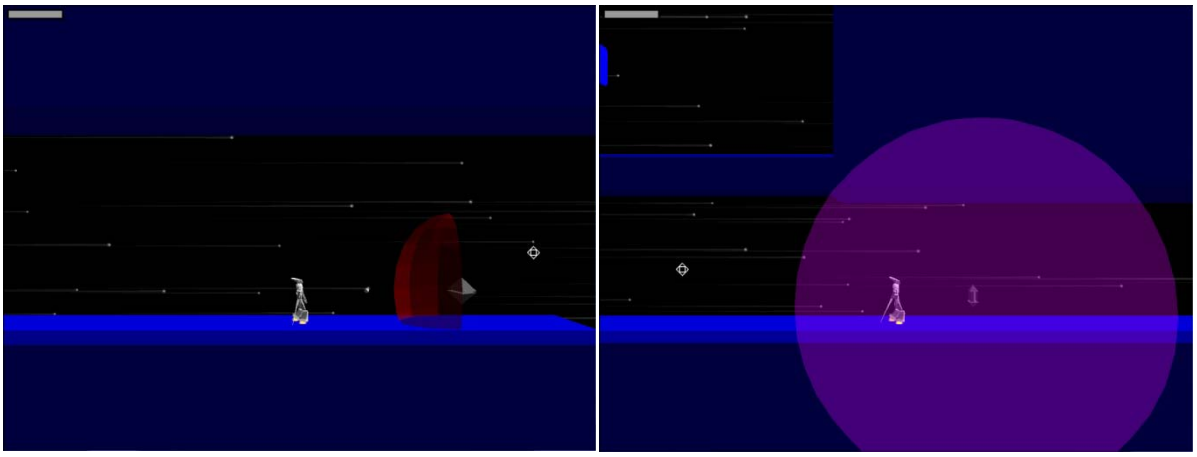

Figure 7. Dematerialized week 2 build.

Week 3

The third week of development was punctuated by the decision to cut the game's original scope from 25 levels including 5 bosses to 10 levels including 2 bosses (one of which is shown in Figure 8). This decision was made to allow the game to appear more polished, and was informed by the realisation of the fact that most players and markers would not ever play long enough to reach the end of the original concept. Many small mechanical changes were made to add polish and accessibility to the game (e.g. respawning delays, level transitions, teleportation approximation), but the largest task of the week was adding animated textures into the game to give it's visuals a more dynamic feel. As the developer had no prior experience with traditional 2D animation, learning how to complete this task consumed a great deal of time, which in retrospect was not justified by the amount of difference it made to the final game.

Other small visual changes were made, including adding a red aura to any dangerous objects and creating a new shape for enemy bullets (as to stand out from the circular background objects). A number of levels were created through experimenting (rather than pre-planning) with the now mostly finished enemy and object catalogue, and the first boss was constructed in order to provide a break from the usual puzzle solving with a more psychomotor based challenge.

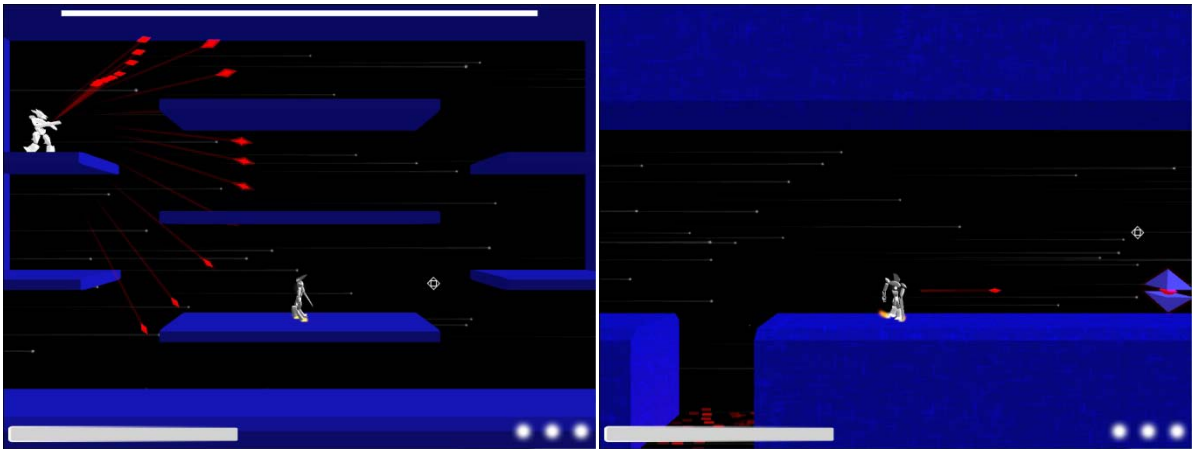

Figure 8. Dematerialized week 3 build.

Week 4

The penultimate week of development saw the completion of all levels but the final boss, and the beginning of more rigorous playtesting and balancing. As in the previous week, the levels were created through the process of placing a number of objects in a scene and experimenting with their layout, allowing the pre-established mechanics to combine with these objects to create emergent challenges. Additionally, some levels were created by updating and tweaking previously discarded concepts, reducing some level of development redundancy.

Aside from level construction, the largest task of the week was the creation of various GUI based elements, including the main menu (shown in Figure 9), intro sequence and prologue level prompts. The greatest difficulty encountered in this task stemmed from the fact that there were no pre-established plans for the game's story (aside from a vague setting), thus giving rise to the difficult task of using the limited GUI to establish a narrative link to the gameplay in a way that would be interesting and motivating to the player.

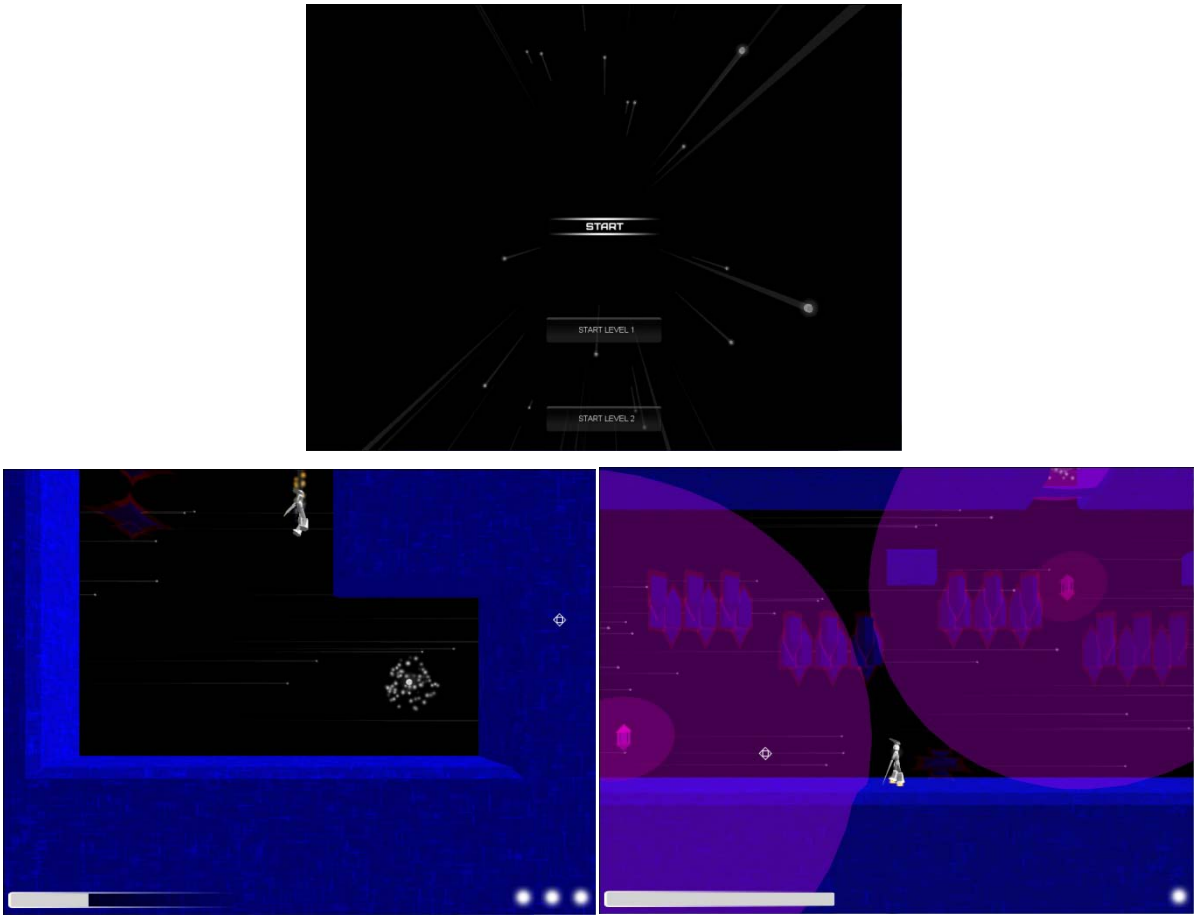

Figure 9. Dematerialized week 4 build.

Week 5

The very first task undertaken in the final week of development, implementing the level select and global time tracking, proved to be problematic and time consuming as a result of a lack of understanding of how to use Unity's GUI systems. The final level that needed to be created was the final boss, which demanded a great deal of time to be spent on tweaking and revising to be a challenge befitting of the end of the game. A lot of time in this week was spent debugging, polishing and balancing, resulting in levels being re-ordered and allowing me to detect and rectify a number of issues which were only encountered in the final compiled game (e.g. pausing by setting the game's timescale to 0 caused a severe drop in performance).

The greatest issue encountered in this week was completely unforseen and had nothing to do with the actual development of the game; the audio designer's computer crashed and destroyed the majority of the pre-constructed music and sound effects. As an emergency measure in order to produce something useable in a short time, the audio designer gathered

and modified some old music and sound effects he had previously created for other projects, while the developer used a freeware sound generator (Bfxr) and Audacity to create some extra sound effects. As Dematerialized was being developed adaptively, what could have been a disaster was quickly and easily managed; the development schedule was adapted to allow some time to create extra sound effects, while the game's theme was loose and abstract enough to seamlessly incorporate a different audio style than planned.

Figure 10 shows a breakdown of number of hours spent on each generic task during the Dematerialized's development, derived from daily hours documented in the development logs.

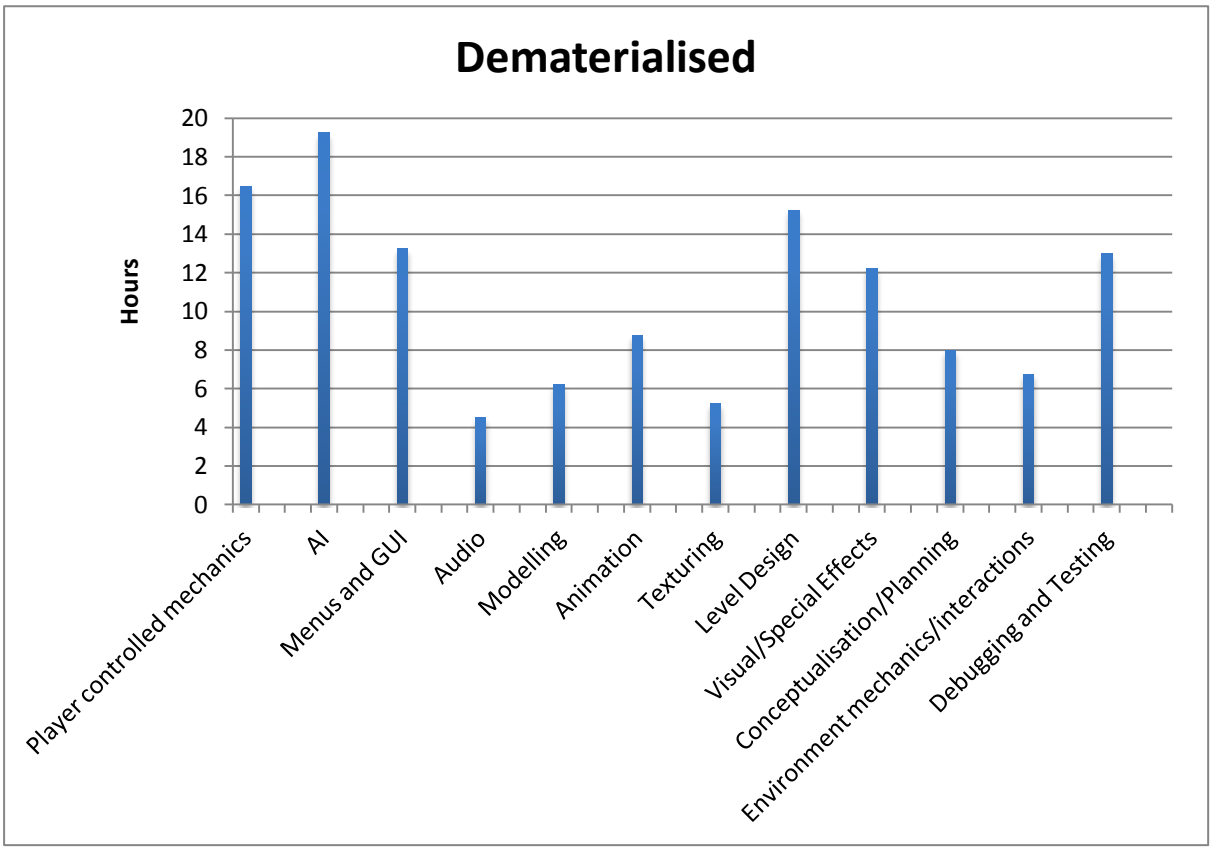

Figure 10. Dematerialized development log task breakdown.

Cursed

The second game was developed according to a predictive methodology, requiring the construction of a set of formal design documents before development began. Due to this, the first week of the game was spent entirely planning, constructing a game design document, a set of level design storyboards and drawing concept art (evidence of this planning can be seen in the appendices). While this initial week would allow for the game's original concept to be firmly embedded in the final product, it effectively took a week away from the actual development of the playable game.

Week 1

Before beginning the five weeks of developing Cursed, the developer had devised a relatively well defined concept of how final game was to look and play. As a result of this, the construction of the game design document (see Appendix 1) and concept art (see Appendix 3) was not very mentally taxing and did not take up a large amount of time. The only aspect of the game design document that needed some extra in-depth consideration was the asset catalogue, as it needed to be large enough to allow for visual and mechanical variety, but small enough to keep the time spent modelling, animating and texturing to an acceptable level.

The most difficult task of this week was by far the level design storyboards (see Appendix 2). Due to the fact that the main challenges in Cursed were planned to come in the form of environment based logic puzzles, constructing the level storyboards essentially became the construction of every challenge the player would encounter in the game. The task of coming up with 14 different logic puzzles within a short one week period proved to be extremely mentally taxing, and while they did not take much time to draw onto a storyboard, the developer found that he could not create more than a few at a time before becoming creatively exhausted.

Week 2

The majority of the first week of actual game development was spent modelling, animating and implementing the player character model, ensuring they move in a sufficiently realistic manner. As the developer was still quite new to organic character modelling and animation, the process of creating the player character was arduous, time consuming, and took a large

amount of research and learning. The final day of the week saw the construction of the first heal object (which failed to animate) and the basic movement AI of the cursed characters (based on raytracing to detect the player), as demonstrated in Figure 11.

After spending week 1 entirely planning, it was a little worrying and frustrating that the majority of week 2 was spend on creating a single (albeit important) asset. Due to the nature of Cursed's predictive development approach, there was the concern that perhaps certain crucial assets would cause unforseen issues that would slow down the creation for the game, and thus cause it to be unfinished at the end of its reduced development time frame. However in retrospect, spending a large time on getting the player character model and controls set up correctly at the beginning of development most likely saved time later in the testing and debugging phases, and thus was overall beneficial.

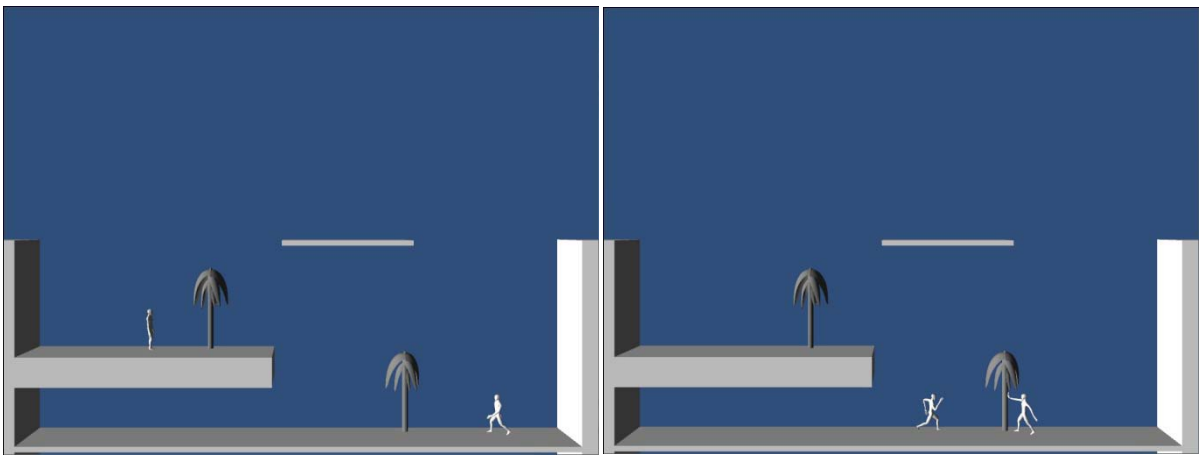

Figure 11. Cursed week 2 build.

Week 3

The most immediately obvious addition to the game in week 3 was a large number of 3D models. The majority of level geometry and heal objects were created, and the player model was fitted with a mask and various jewellery (see Figure 12). Another important visual milestone was the beginning of texture and material creation.

A number of final gameplay mechanics were created, including cursed healing detection and response (demonstrated in Figure 12) and player-level interaction. Player-level interaction proved to be one of the most difficult mechanics to implement, creating unforseen problems with gate movement when a lever is pulled and requiring a multitude of player movement overrides when attempting to climb a ladder. These issues were resolved with perseverance

and a few work around methods, but the ledge grab/hand mechanic proved too troublesome to implement, and was cut for the sake of keeping development time within the 5 week period. This design decision was perhaps the biggest made during active development, and caused more time to be spent re-working a number of level designs in later stages of development.

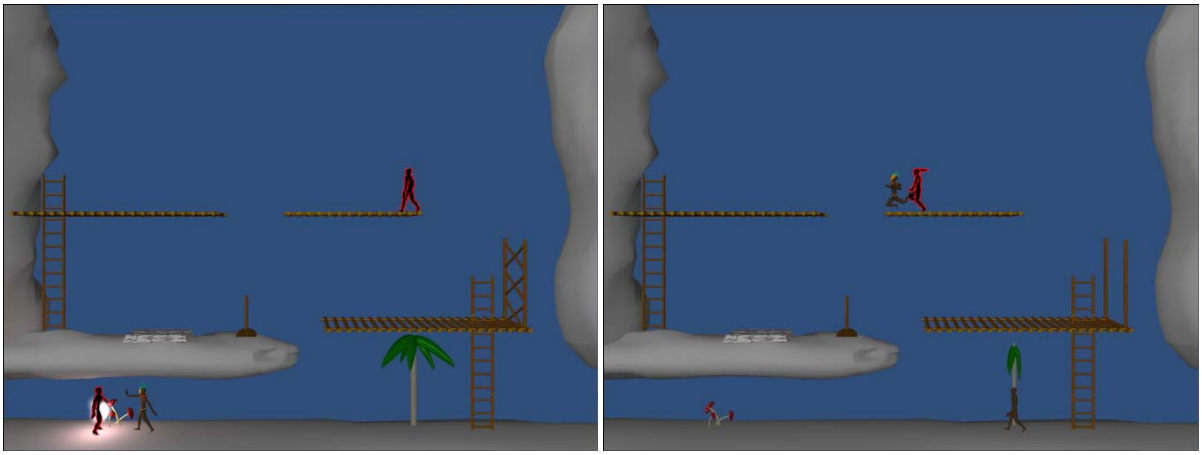

Figure 12. Cursed week 3 build.

Week 4

The fourth week of development saw both the beginning and end of regular level development. Every one of the 14 main levels were created and populated with pre-constructed objects, based on the pre-constructed level design storyboards. A number of alterations had to be made in light of the decision to remove ledge grabbing and hanging, but overall the challenge and general structure of each level remained the same. It was this phase of level development that most demonstrated the virtues of predictive development; what was a difficult, time consuming, trial-and-error task in Dematerialized's development became an easy, relatively quick one.

Aside from a number of small playtesting tweaks made as a requirement of getting every level to function, the second most obvious addition to the game in week 4 came in the form of more visual improvements (Figure 13). The rocks were given a texture created in Photoshop, and the background was populated with rubble, a beach plane made in Maya, pre-constructed water from Unity and a sky plane painted using a custom Photoshop brush, all created with relative speed and ease. The ability for the player to use their own health to heal cursed was implemented, and in turn GUI health bar functionality was added.

Figure 13. Cursed week 3.5 build.

Week 5

By Cursed's final week of development, the vast majority of in-game features were already implemented, including assets and mechanics. The front end and pause GUI were easily constructed through the adaption of scripts created for Dematerialized, allowing them to be quickly and easily implemented (shown in Figure 14). The opening scene was created with simple GUI commands, but the final level and ending scene proved to be problematic in creating a sense of weight and consequence. A large amount of time was spent on getting the final level right, as it was a particular scene which had been clearly visualised from the game's inception, and re-enforced the entire theme and message that was attempted to be communicated through the game's mechanics.

As in the end of many game development cycles, a large amount of time was spent playtesting, polishing and debugging. Unfortunately, due to the use of a predictive development methodology for this game, there were a large number of small issues and bugs that were identified. In this late stage of development only the most problematic bugs could be removed in time; less impactful ones had to be selectively left in for the sake of completing the game on time. Based on playtesting results, the decision was made to create an extra, simplified tutorial level, as the first tutorial proved a little too complex when completely unfamiliar with the game's mechanics (there was little way to tell this when originally storyboarding on paper). Finally, the audio was easily implemented, mostly through the re-use of scripts created in Dematerialized (especially the music manager).

Figure 14. Cursed week 4.3 build.

Figure 11 shows a breakdown of number of hours spent on each generic task during the Cursed's development, derived from daily hours documented in the development logs.

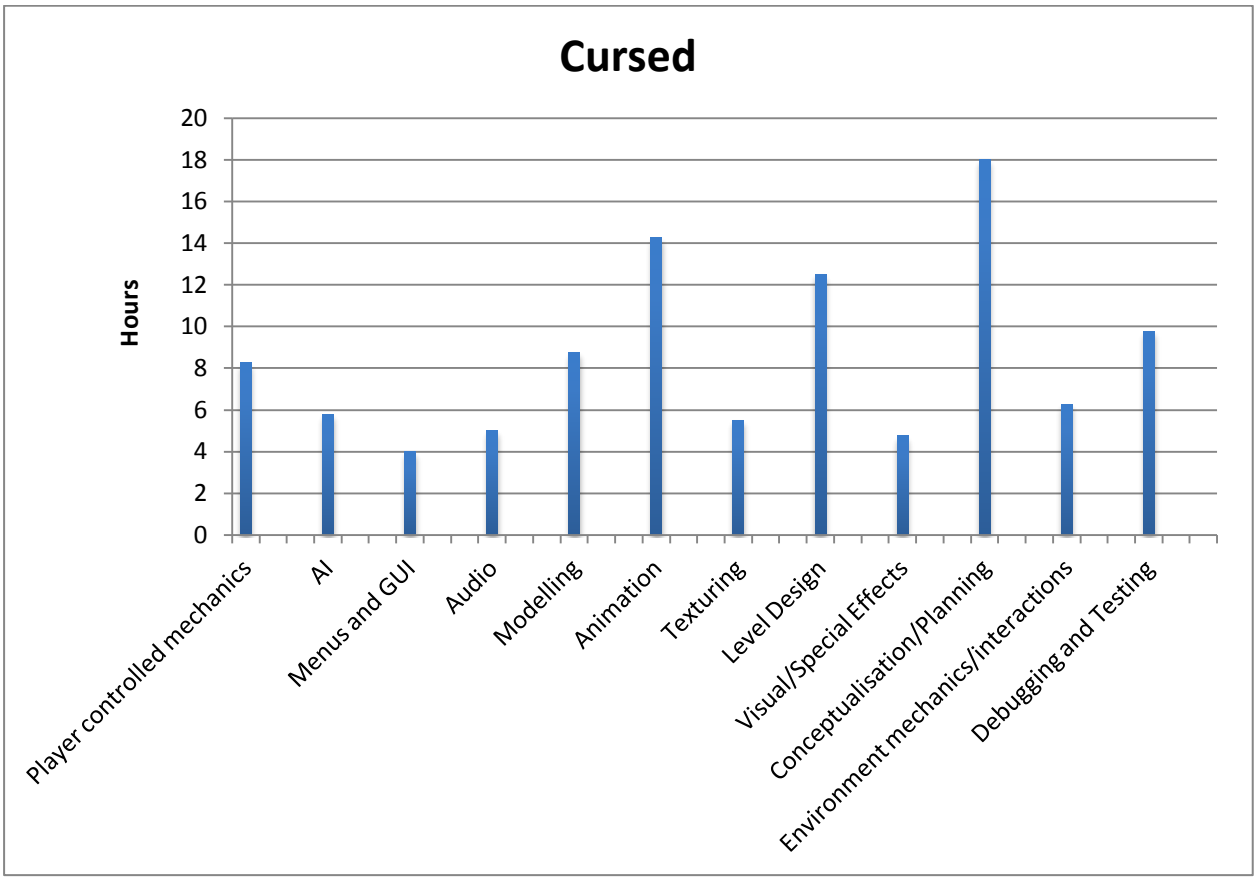

Figure 15. Cursed development log task breakdown.

Playtesting

The following graphs and tables summarise the data gathered during a series of playtests of the two games Dematerialized and Cursed, using the playtesting method outlined by Davis, Steury & Pagulayan (2005) in the literature review. A total of 60 participants took part in the playtests, whom were all requested to play both games for a minimum of twenty minutes and then fill out a questionnaire on each. Each questionnaire consisted of a section asking the participant to respond to a number of statements using a likert scale, a section asking them to rate aspects of each game out of 10 (10 being the highest), and a section asking them to identify (in their own words) areas they liked, disliked, and thought needed improvement. A sample questionnaire used for both Dematerialized and Cursed is available in Appendix 5. Of the total number of participants, seven were selected to be observed and asked questions while playing for an extended period of time (approximately an hour for each game). These participants were asked to verbalise their thoughts as they played, with their behaviours being observed and recorded based on Consalvo & Dutton's (2006) four areas of qualitative game analysis.

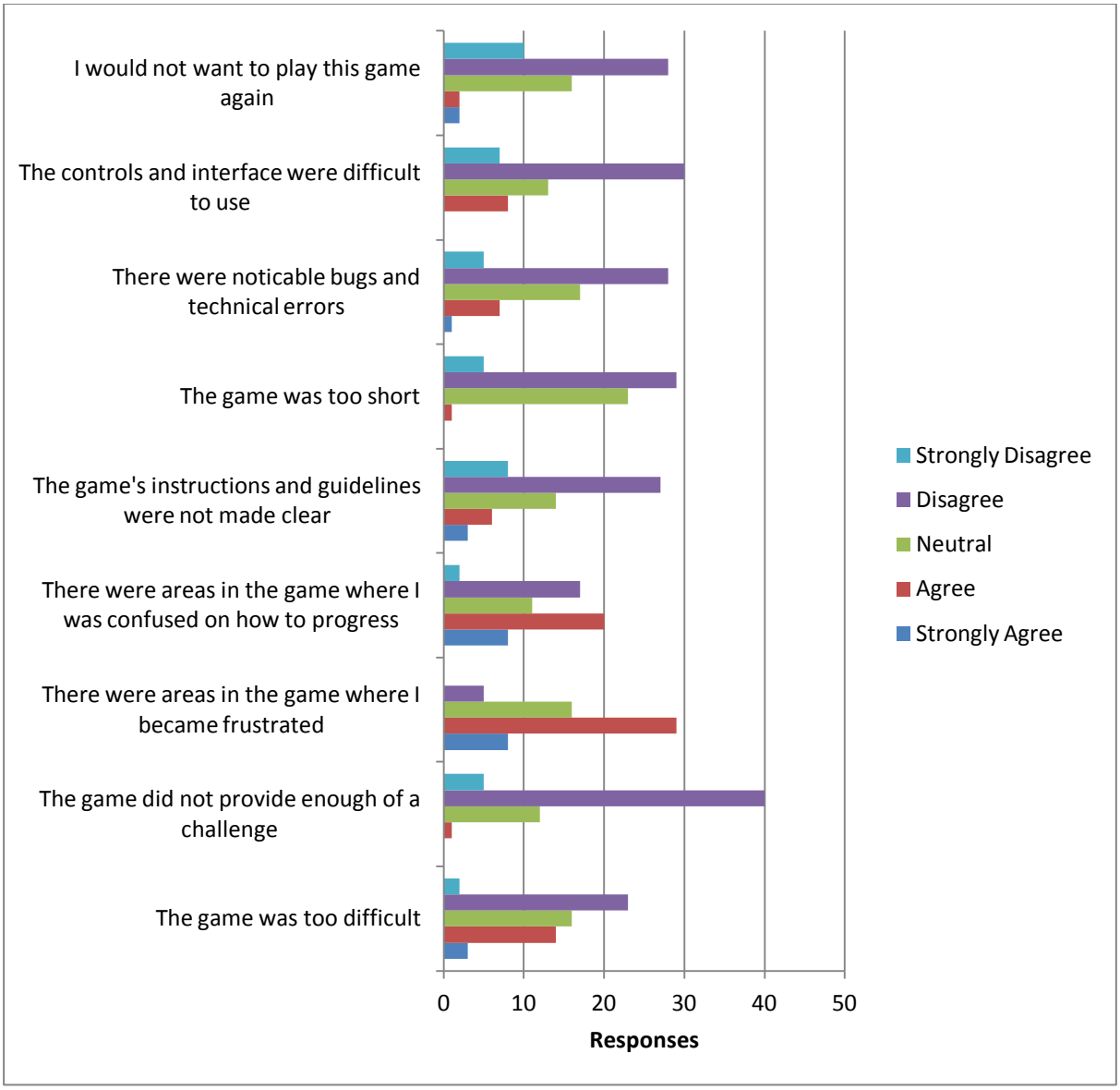

Figure 16. Cursed feedback likert scale.

As shown in Figure 16, the results of the likert scale feedback from Cursed show that playtesters generally disagreed with the majority of the negative statements, yet however agreed that the game contained a number of frustrating and confusing areas. Tables 3 and 4 show the ratio of agree to disagree, and was constructed by finding the number of participants that ‘strongly agreed’ or ‘agreed’ to each statement, then dividing this number with the number of participants that ‘strongly disagreed’ or ‘disagreed’ to each statement, in order to obtain a comparative ratio (1 > indicates general agreement, 1 < indicated general disagreement). The higher the ratio number, the greater the amount of playtesters that agreed with the statement.

Table 3. Cursed feedback likert ratios.

	Agree	Disagree	Ratio
The game was too difficult	17	25	0.68
The game did not provide enough of a challenge	1	45	0.022222
There were areas in the game where I became frustrated	37	5	7.4
There were areas in the game where I was confused on how to progress	28	19	1.473684
The game's instructions and guidelines were not made clear	9	35	0.257143
The game was too short	4	34	0.117647
There were noticable bugs and technical errors	8	33	0.242424
The controls and interface were difficult to use	8	37	0.216216
I would not want to play this game again	4	38	0.105263

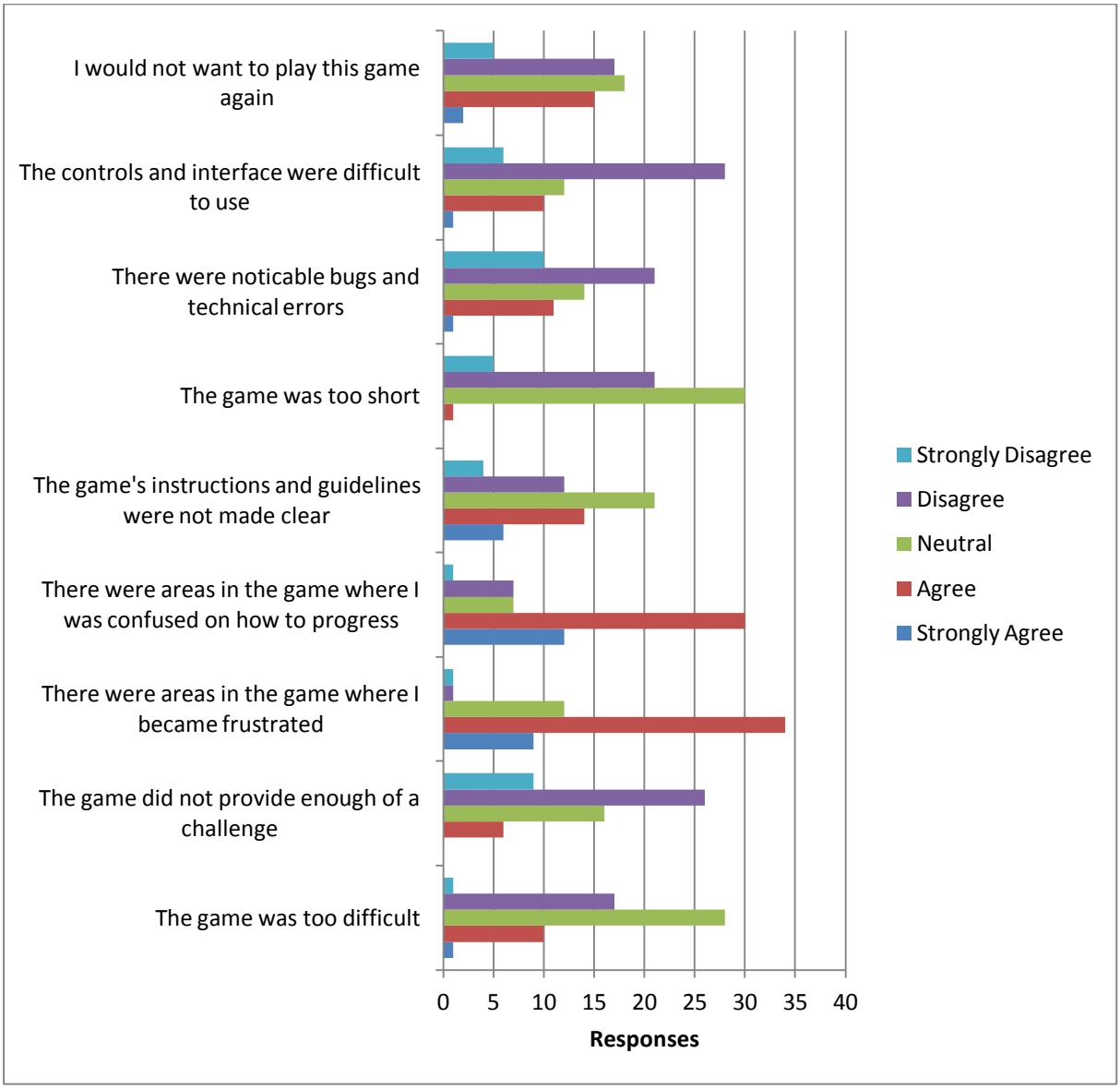

Figure 17. Dematerialized feedback likert scale.

While the likert scale feedback from Dematarialized is similar to that of Cursed, a comparison of Table 3 with Table 4 shows that playtesters generally considered the game to be more frustrating, confusing, and lacking sufficient instructions or guidelines. Specifically, the statement “There were areas in the game where I became frustrated” resulted in an agree ratio in Dematerialized of 21.5 versus a ratio of 7.4 in Cursed. The statement “There were areas in the game where I was confused on how to progress” resulted in agree ratios of 5.25(Dematerialized) versus 1.47(Cursed). The statement “The game's instructions and guidelines were not made clear” resulted in agree ratios of 1.25 (Dematerialized) versus 0.26 (Cursed).

Table 4. Dematerialized feedback likert ratios.

	Agree	Disagree	Ratio
The game was too difficult	11	18	0.611111
The game did not provide enough of a challenge	6	35	0.171429
There were areas in the game where I became frustrated	43	2	21.5
There were areas in the game where I was confused on how to progress	42	8	5.25
The game's instructions and guidelines were not made clear	20	16	1.25
The game was too short	1	26	0.038462
There were noticable bugs and technical errors	12	31	0.387097
The controls and interface were difficult to use	11	34	0.323529
I would not want to play this game again	17	22	0.772727

Figures 18 and 19 graph the distribution of scores assigned by playtesters to four general aspects of each game. For every scored game aspect, playtesters generally assigned a slightly higher rating to those in Cursed than those in Dematarialized, as shown in Table 5.

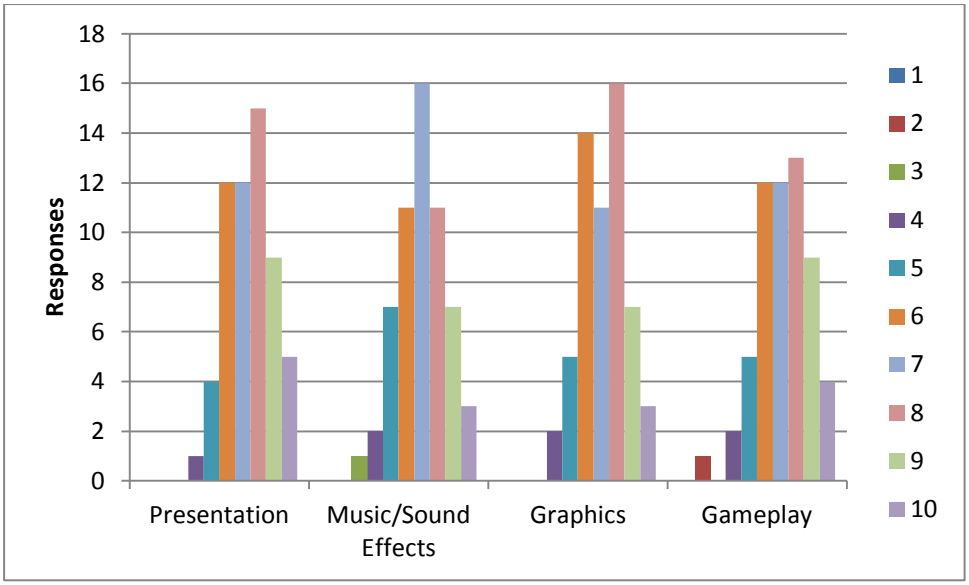

Figure 18. Cursed feedback scores.

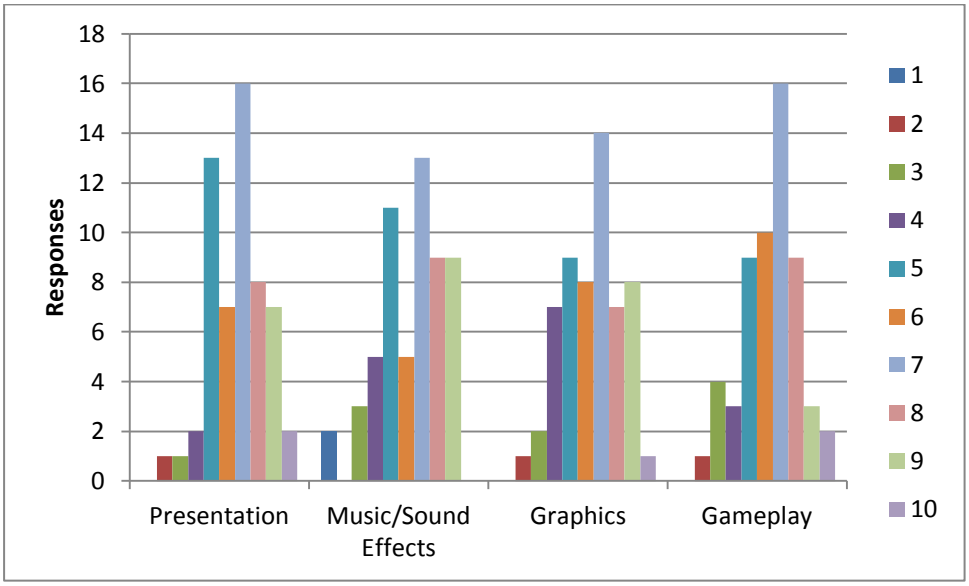

Figure 19. Dematerialized feedback scores.

Table 5. Average feedback score comparison.

	Dematerialized	**Cursed**
Presentation	6.649	7.431
Music/Sound Effects	6.316	6.983
Graphics	6.404	7.155
Gameplay	6.301	7.172

The following are areas in Dematerialized and Cursed that playtesters commonly enjoyed or disliked. These areas were determined by analysing the section in the questionnaires asking playetesters to identify areas in each game that they liked, disliked, and thought needed improvement. Areas that are listed below are those that appeared multiple times in similar written responses.

Common areas enjoyed in Dematerialized:

- Presentation and visual cohesion.
- Teleportation mechanic.
- Music.

Common areas disliked in Dematerialized:

- The bosses.
- Difficulty and timing challenges.
- Lack of instruction and explanation.
- Reliance on player experimentation and intuition.
- Lack of in-game instructions/tutorial.
- The controls – multitasking between mouse and keyboard.

Common areas enjoyed in Cursed:

- Logic puzzles and capacity for pre-planning.
- Graphical fidelity.
- Cursed enemies – Tension and sound effects.
- Level of challenge and linear difficulty progression.

Common areas disliked in Cursed:

- Specificity of certain mechanics (e.g. distances from healObjects, ladders).
- Difficulty and complexity.
- The controls and lack of menu functionality.
- Repitition of background and scenery.
- Difficulty of tutorials, hidden or unexplained mechanics.
- Lack of instruction on how to use the health mechanic.

Discussion

The way in which a game is developed can have a significant effect on both its commercial and critical success. In order to construct a set of recommendations and guidelines for independent developers drawing on both adaptive and predictive methodologies, the implications of specific aspects of these development approaches must first be understood. Investigation into the ways in which Dematerialized and Cursed were affected by their adaptive or predictive development archetype (respectively) have been split into three main areas of study: design, development, and final player experience. Through these three areas this study identifies specific ways in which each game's development methodology affected its content and overall structure, its development process and ease of construction, and the way in which players experience and respond to its finished release. It must be noted that the finished game releases discussed in this paper and used in playtesting have since undergone a small number of changes and bug fixes (lists of which are available in Appendix 4).

Design Reflection

In order to analyse the design, meanings and promoted behaviours in Cursed and Dematerialised as a result of their development methodology, a reflection of the design of the games has been conducted following objective criteria for game analysis proposed by Consalvo & Dutton (2006). While the results of these games being analysed by their own developer may contain sub-conscious bias, they have been triangulated with developer experience and playtesting feedback in order to determine specific areas in each game that contain contrasting meanings and gameplay.

Object Inventory

Dematerialized contained very little in the way of objects that could be interacted with (teleportation jammers, shields and end level goals). Each object provided a self-contained function and in most cases existed as a hindrance to level progression that could only be overcome in one particular way. Conversely, the objects in Cursed were more numerous and in most cases were there to aid the player in the completion of every level (e.g. the heal objects, levers and ladders); they were interrelated functioned together as a cohesive network to solve puzzles.

The difference between the objects in each game can be directly attributed to the methodology used to create them. The objects in Dematerialized were not planned or developed together, and were not created in tandem with the design of the levels or game mechanics. Thus each object had to function as an independent module, based on the fact that game elements could be introduced or removed throughout the entire development process. The objects in Cursed were planned and designed at the same time as each other, prior to the commencement of active development. This resulted in them becoming interdependent, due to the fact that every game element was known and quantified at the time of their design, allowing their functions to rely on these external elements.

Interface Study

The primary difference in interface between Dematerialized and Cursed lies in the way they obscure information from the player. As only small portions of Dematerialized's levels could be views on screen at any given time, level information was obscured from the player, forcing them to explore and resulting in unforseen situations. Conversely, Cursed's levels were viewable in their entirety at all times, allowing the player to plan their approach in advance and reasonable predict the outcomes of any given action.

The way in which the design methodology affected the interface of each game can be related back again to the way in which they were planned. As Cursed's levels were entirely pre-planned by hand they achieved a level of cohesion and interrelation (puzzle elements were linked and functioned in tandem with each other) that was not present in Dematerialized, allowing a measure of complexity that allowed each level to still be challenging if its entire structure was known by the player. However, the fact that Dematerialized's levels were planned iteratively and experimentally allowed them to have greater breadth and variety at the expense of complexity and cohesion between elements, requiring the obscuring of information in order to keep the player engaged through a sense of discovery.

Interaction Map

Dematerialized contained only NPCs that were hostile to the player, while Cursed also contained NPCs that the player had to protect and assist. Overall, the function of the NPCs in Cursed was far more complex and important than that in Dematerialized. They embodied both the main goals and challenges of the game, rather than simply serving as a hindrance to the player's progress. However, while Cursed only contained one type of NPC with only one

major way to interact with them, Dematerialized involved numerous who could be confronted or avoided at the player's discretion.

The NPC interactions in Cursed were inherently tied to the game mechanics, setting and narrative, as a result of the enemies being designed into the game at the point of conceptualisation. However, due to the adaptive nature of Dematerialized, NPC interactions that were planned from conceptualisation ended up being scrapped from the game (see week 2 in Dematerialized's development summary), and thus the enemies that were later added were not created in tandem with the mechanics. This resulted in them becoming less essential to the overall gameplay, which allowed a greater variety of enemy behaviours but reduced the depth of interaction that the player could have with them (as they were not intimately related to the main teleportation mechanic).

Gameplay Log

While both games could be considered to involve puzzle challenges, Cursed's design generally encouraged logic and planning due to the structure of its puzzles and repetitiveness of game elements, while Dematerialized encouraged a greater amount of control precision and experimentation due to the fact that levels were obscured and challenges were varied. The variety and independence of individual elements in Dematerialized resulted in a greater potential for the player to become confused, disoriented or frustrated; new elements and situations were frequently presented which demanded the player to adapt their style of play accordingly. Conversely, Cursed's overarching gameplay maintained a very similar structure throughout its duration, allowing players to build upon their initially established approach to gameplay with a certain level of repetitiveness.

The variance and repetition inherent in Dematerialized and Cursed's gameplay respectively was a direct influence of the manner in which they were developed. Due to its linear development approach, Cursed grew from a single, simple but thoroughly planned gameplay mechanic, allowing all subsequently developed elements and stages of development to build upon this method of interaction with minimal divergence. This resulted in a more predictable and reliable path of gameplay progression, as evidenced by every level and challenge being structured around luring enemies into areas they could be healed. Dematerialized however evolved over a series of iterations and revisions, allowing new methods of gameplay to be introduced that would not disrupt its core teleportation mechanics due to game element

independence and modularity. Examples of these methods of gameplay include progressing by using momentum, deflecting bullets, predicting boss movement patterns and teleporting accurately.

Development

Perhaps the most immediately obvious difference between the development of the two games is the fact that Dematerialized had five weeks of active development (asset creation, programming, etc.), while Cursed only had four. As Cursed was developed under a predictive methodology, and thus demanded some form of initial planning and formal documentation, the first week of its 5 week timeframe was spent planning and documenting. During this phase all aspects of the game were planned and accounted for, from mechanics to levels. The amount of hours worked in this particular week were far lower than usual (13.5 hours, derived from development logs) which can be attributed to the creatively intense nature of the tasks performed. Unlike active development, where important decisions only need to be made every so often and ideas can quickly be tested and prototyped, every step of the planning phase required careful consideration and cross-referencing with already defined features of the game. Due to Cursed's nature as a logic puzzle game, the level designs needed to be mentally playtested, ensuring that they were challenging and balanced without the aid of the immediate feedback usually received from real prototype playtesting.

In comparing the levels of Cursed and Dematerialised and how they were developed, certain effects of the predictive and adaptive archetypes could be determined. While Dematerialized introduced new settings and different types of challenges throughout its levels, Cursed's setting and challenge types remained relatively unchanged from start to finish. This is a direct result of the way in which these levels were planned. The levels in Dematerialized were planned literally from the beginning to the end of its five week development period, and thus were influenced by the particular development phase the game was in and any new ideas the developer formulated as a result of iterative playtesting. On the other hand, Cursed's levels were all planned within less than a week, thus allowing less time for new ideas and resulting in a game with a more consistent challenge type. It should be said that having planned Cursed's levels in advance cut down the difficulty and development time of creating the levels in game (12.5 hours), when compared to those of Dematerialized (15.25 hours, shown in Figures 10 & 15).

In adherence to a predictive development methodology, the development of Cursed was much more planned and consistent than that of Dematerialized. This is most obvious when comparing Cursed's game design document timeline with its actual development process; tasks were undertaken in a specific order that was considered before active development had begun. When contrasted to Dematerialized's somewhat turbulent development, the planning and consideration put into Cursed's development resulted in a reduction of unforseen problems and circumvented the production of redundant or unused game elements (assets, mechanics, etc.)

The integration of iterative playtesting into Dematerialized's development as part of a adaptive methodology afforded greater time to be spend debugging and playtesting (as shown in Figures 10 & 15) , reducing the chance of bugs and issues being forgotten or put off till later in favour of other features. This resulted in a reduction in solvable number of bugs (a few minor unfixable problems with the Unity engine remained) through the adaptive method, but also took time away from other tasks, increasing development time and reducing the amount of content that could be made. Those put off under Cursed's predictive model were only those that did not hinder the game, but were eventually left in due to time constraints, diluting the challenge in certain areas.

A large difference between the predictive and adaptive approach was the compromise of the game's initial vision and concept. While Cursed stayed almost identical, Dematerialized changed radically, almost shifting genres. This was as a response to lessons learnt in development, upon the realisation that the initial concept was too complex in scope for a 5 week period. This caused dilution and compromise (not always acceptable), but allowed the game to be completed under a strict timeline to a satisfactory degree. Conversely in a predictive approach, if planning gets out of hand an entire game can go far beyond schedule or entirely collapse during its own development. Making changes to a game concept while developing predictively is much more difficult than when using an adaptive approach due to pre-planned feature interdependence.

As shown in Figures 10 & 15, player controlled mechanics and AI were a greater focus in Dematerialized's development (16.5 and 19.25 hours respectively), while modelling and animation were instead focused on in Cursed (8.75 and 14.25 respectively). In turn, the former game possessed a greater variety of gameplay options and challenges (through the

players choices on how to interact with the AI) while the latter achieved a higher level of graphical fidelity. During Dematerialized's development cycle a greater amount of time was spent on visual special effects (12.25 vs. Cursed's 4.75 hours). Although this extra time resulted in a number of playtesters identifying the game as having a more consistent visual style, much of it was spent tweaking small effects (some of which never made it into the final game, such as animated textures). The nature of Dematerialized's adaptive development allowed time for this visual experimentation, iteration and testing of styles, but caused redundancy and often resulted in only a small visual improvement for the amount of hours put in.

As shown in the development section of this paper, Dematerialized's adaptive approach allowed problems and setbacks to be taken into account during development and the game to be revised appropriately. This was particularly evident when the game's audio files were lost, with the development cycle being adapted and different tasks re-prioritised in order to allow time for the last minute creation and implementation of new audio. By its very nature, the adaptive development methodology allowed the game to easily adapt to issues and problems encountered during active development. In contrast, during Cursed's development it was discovered that a certain game mechanic (ledge hanging and climbing) would have to be removed in order to meet the five week deadline. Because of the predictive nature of the game's development methodology a number of other game elements were already dependent on this mechanic, and thus removing it proved to be a difficult and time consuming task due to the fact that the development approach was not geared towards adaption.

Player Experience

Through playtesting observation and qualitative feedback from the questionnaires, it was found that Dematerialized demanded a level of attention, engagement and work from the player that was not reinforced by its story and setting. The player was given small motivation to invest in the game, aside from enjoyment of the mechanics themselves. Generally, those who put in the effort and investment (without any real evidence of return) enjoyed the game better than Cursed, but most found it too difficult to get into and became frustrated.

The majority of playertesters who claimed to be experienced with video games preferred Dematerialised, claiming to have enjoyed its variety, complexity, simpler and more consistent visuals, and the exploration of the teleportation mechanic. Less experienced players tended to

prefer Cursed, providing reasons such as its more consistent gameplay style, its narrative and setting, its increased graphical fidelity and the consistency and predictability of its challenges.

On the whole, when asked about Dematerialized's story playtesters agreed that it was too minimalist and abstract to become involved in, and consequently did not realise what was going on in the game's narrative. The gameplay and narrative were recognised as being developed separately, and it became obvious that the story, theme and setting came second to the game's mechanics, which frustrated many of the less skilled players. When asked to specify areas of improvement in the questionnaire, playtesters often suggested that more story segments or prompts be placed between levels, with many feeling that while the story didn't detract from the game, it did not add much or contribute to player motivation.

The predictive approach used by Cursed was shown to be more appropriate to position narrative or setting as one of the key attractions of the game. The game's development methodology allowed the narrative to be worked into the entire development process and mechanics from the very beginning. When asked how the narrative differed to that of Dematerialized, playtesters recognised it as being re-enforced through the gameplay and easy to understand. It was difficult to maintain this level of focus on narrative and setting when using adaptive to develop Dematerialized, which resulted in many players feeling indifferent to the game's story and setting. Conversely, playtesters who had more experience with games appreciated the mechanics focused approach of Dematerialized afforded by its adaptive development process, highlighting the exploration and evolution of the initial mechanics as key gameplay motivators.

When asked about the setting and theme of the two games, playtesters identified Cursed as having greater cohesion between its gameplay and these elements, a result of all being planned at the same time. In the same sense, most playtesters acknowledged the fact that the game's concept was essentially taken to its limits; it could not be extended further, and in some cases the gameplay was too similar and somewhat boring by the end. Because of the way in which gameplay, setting and theme were intertwined and pre-planned, many features could not be added or removed once development had begun, as a result of each element relying on each other to function correctly. This resulted in the game becoming somewhat predictable in its presentation and challenges, which some playtesters appreciated for the consistency and other disliked for the repetition.

Dematerialized was identified as having more variety and variation between levels, allowing many areas to play and feel differently. In turn, many playtesters regarded the game as more frustrating and confusing (see Playtesting section and Tables 3 and 4) as the challenges were constantly in flux, requiring the player to devise brand new approaches to many levels rather than building off pre-established ones. These factors were influenced by the game's adaptive development approach, which allowed the introduction of new ideas over the entire development cycle and the flexibility to adapt entire sections of the game based on lessons learnt during development. Even though less time was spent on asset creation in Dematerialized, many playtesters enjoyed its presentation and visual cohesion (see common areas enjoyed and disliked in the Playtesting section) – different colour palettes were able to be tested and certain elements could be compromised to fit in with others with due to the abstract nature of the setting.

As shown in Tables 3 & 4, playtesters considered both games to contain roughly the same number of bugs. Those in Dematerialised were generally identified as a minor annoyance that did not greatly affect gameplay, while those in Cursed usually helped the player (and made the game easier), but occasionally broke the game and made it unable to complete the game fully. The way in which bug removal was carried out was affected by the development methodology of each came. The bugs that remained in Dematerialized were simply ones that the developer failed to detect during testing. The bugs that remained in Cursed were generally the ones that were deemed least detrimental to the game, as not enough time was allocated to testing and debugging in order to remove them all within the 5 week development period.

Overall, Dematerialized was rated lower than Cursed in the majority of areas. (see Tables 3, 4 & 5). This can be attributed to a number of factors, the first of which is simply the fact that it was developed first, and thus Cursed benefited from the skills gained by the developer through spending 5 weeks using Unity, Maya and Photoshop (a limitation of this project). The second factor is the fact that Dematerialized was intentionally designed to be a harder game for more experienced players, which the majority of playtesting participants were not. The third and most important factor is that, as Dematerialized was designed adaptively and thus not-pre planed, many sub-conscious assumptions were made based on the developers subjective knowledge of the game (e.g. a blue texture denotes an indestructible enemy), and thus caused players to become frustrated, angry and less charitable when the time came to

rate the game. These assumptions did not materialize in Cursed due to the fact that game elements were given more conscious and planned consideration. Cursed's simplicity as a result of introducing a defined scope at the start of development proved to be beneficial, while Dematerialized's complexity as a result of a variety of different, independent elements being introduced at different development stages proved to hinder its playability and accessibility.

Recommendations and Conclusions

In order to properly provide a set of development recommendations and guidelines for independent developers, the specific characteristics intrinsic to all independent video games must be determined. For the purposes of this study, independent video games will be considered to possess the following attributes, further detailed by Kerr (2006) and Stevens (2009) in the literature review:

- Testing ground for new or innovative ideas.
- Low financial investment and risk.
- Fast development cycles, compared to those funded by a publisher.
- Financed and marketed without a publisher.
- Digitally distributed.
- Targeted towards a specific audience or niche.
- Released with a small user install base.

The choice between a more adaptive or predictive approach must be mediated by many factors: the development team; the kind of game they want to make; and their target audience. However, there are very few instances in which a developer should adopt an entirely predictive or adaptive approach, and rather should use a mix of the two.

Many playtesters felt as if the narrative in Dematerialized was 'tacked on' (it was added after active development began) and did not benefit the game as a whole, while that in Cursed successfully reinforced and motivated gameplay. As such, if narrative is to be an integral part of a game, it should be worked into the gameplay and not exist as a separate experience. In order to ensure this, the **narrative should be predictively constructed; the developers should know it in its entirety before active development begins, ideally creating storyboards, scripts or tangible plans to some extent.**

As asserted by Stevens (2009), in almost every instance prototyping and iterative playtesting should take place through multiple stages of development, in order to ensure that every aspect of the game is enjoyable and functional. While iteratively playtesting in Dematerialized's development consumed a large amount of hours, this was offset by time saved though avoiding a number of problems later in development. **Independent developers should**

playtest constantly throughout development, ensuing that each element is functional before other game elements become reliant on it. Doing so can cut down on the number of things that need to be tweaked or changed if bugs are discovered by rectifying issues before they become interdependent with other aspects of the game.

Stevens (2009) argues that a game should be planned around the team's strengths. During the development of Dematerialized, a lack of such planning became evident when the game concept had to be significantly altered due to a lack of time and skill. As evidenced by Grossman (2004), instances like this in which game features have to be cut due to a lack of ability, a misjudgement of scope, or unrealistic ambition are common in start-up independent development teams. In this sense, it is recommended **that independent teams should at least partially conceptualize and plan the core features of their game before development begins. The game's scope should realistically be considered in terms of the team's abilities and first and foremost focus on delivering a finished, playable game.**

One obvious difference between predictive and adaptive development methodologies is their approach to documentation. In independent development, documentation should vary depending on team size and cohesion; **if a team is large or split between locations, common reference documentation that all members can defer to is near essential.** Grossman (2003) asserts that there should be a central creative source in development teams for members to defer to, to avoid inconsistencies and misconceptions. As experienced in the development of Dematerialized and Cursed, documentation can cause some areas of the development process to proceed much smoother (in Cursed's case, level design), but does take time and is a creatively taxing job, which could be problematic if deadlines are involved (which in turn help stop feature creep). **The need for documentation when an independent game is being developed by one person is dramatically reduced, but it should still be constructed if the game is to have cohesion of narrative, theme and setting with gameplay.**

The high levels of player frustration and confusion in both games (shown in tables 2 & 3) can be attributed to the fact that little to no external playtesting was conducted during active development; the mechanics and design demanded a level of familiarity that only the developer possessed. As re-enforced by Steven (2009) & Nutt (2011), it is recommended that **independent developers ideally implemented and use external playtesting in different**

stages of their game's development, in order to ensure that all aspects of the game are comprehensible by outer parties. Doing so can provide an outlet for the developers to defamiliarise themselves with the game and assess it from a foreign perspective. Such playtesting does not necessarily have to be formalised as it was in this study, but some level of external feedback should be drawn in order to ensure the game can be understood and played by outside parties, preferably from members of the chosen target audience.

As shown through the discussion of the qualitative playtesting feedback in the previous player experience section of this paper, a predominantly predictive development approach may encourage an emphasis on pure game mechanics, and thus will be more likely appreciated by experienced game players and those who enjoy exploration, experimentation and can be intrinsically motivated to play a game from the action of playing itself. Conversely, an adaptive approach is more suited to creating a cohesive setting, theme and narrative, allowing elements other than the gameplay itself to become the primary player motivation, and thus may be more suitable for players who are less familiar with video game mechanics. **When selecting a development approach independent game developers must carefully consider their target audience, considering their motivations for playing games, skill level, prior experience and attention span.**

Kazdal (in Nutt, 2011) points out that if an independent game is to be commercially successful, it must be demonstrated to potential buyers and marketers early and often. Following this logic, independent developers looking to make a profit should aim to have a playable and functional prototype of the game running as early as possible. There should be small sections which can be shown off that appear 'finished', even if they are only a small slice of the entire game. As shown through the development of this project's two games, the adaptive approach of Dematerialized allowed a semi-finished prototype of a game level to be completed by week two, while such a prototype was only playable by week four of Cursed's development. In order to construct such prototypes, it is recommended **that independent developers employ a measure of predictive planning to ensure that their prototype is representative of what prospective buyers can look forward to in a final release.** However**, an adoption of an adaptive approach to prototype iteration is crucial to constructing a 'finished' slice of the game while other game elements are still being developed.**

As shown in Dematerialized's development summary, an adaptive approach can cause game features created during development to be culled from the final build, leading to redundancy. Cursed's predictive development cycle reduced instances where such culling was necessary, but made it much more difficult to do when it was, due to feature interdependence and development linearity. **If working adaptively, independent developers must be prepared to make hard decisions about getting rid of game elements they have spent hours creating, due to the experimental and iterative nature of adaptive development.**

Grossman (2003) asserts that start-up developers should restrain their ambitions and define their project's limits early on. Focus must be on playability, polish and getting a game finished, rather than sheer bulk. In this sense, some kind of restraints should always be agreed upon by developers before active development begins. The lack of this in Dematerialized caused redundancy, and increased the amount of time spent debugging (figure 10). This finding suggests that **independent developers should build their way up creating games they are sure they can complete, constructing mechanics and assets at a level of detail and complexity in line with their skills.** However, Kerr (2006) asserts that independent developers should be free to try new methods, ideas and innovations, and thus should leave room for revisal and addition to their concepts once they are sufficiently executed, suggesting all development cycles should involve a form of iterative analysis and re-design.

When it comes to gameplay features and mechanics, the playtesting feedback gathered during this study suggests that predictive approaches encourage cohesion, while adaptive approaches encourage variety. Ideally independent developers should aim to find an acceptable medium between variety and cohesion in order to provide mechanics that are compelling but do not become repetitive, suggesting that a baseline or standard should be constructed that all features adhere to, with allowance for permutation. In this sense, **the 'hooks' or core mechanics of a game should be few but interactive, allowing for combinations of such mechanics to give rise to emergent gameplay and challenges.**

Through the consideration of the multitude of effects adaptive and predictive development can have on the design, development and final player experience of a game, this study comes to the conclusion that there can be no one set model that is most effective for all independent game development projects. Instead, projects must be considered on a case by case basis; the

recommendations and guidelines in this paper aim to better equip the independent developer to select their own unique development approach, drawing on a mix predictive and adaptive elements based on their resources and game concept. Rather than assessing the usefulness of adaptive and predictive approaches to a specific subset of the video games industry as this study has, it is suggested that future research in this field could investigate the application of game development methodologies to specific gameplay types and genres. Researches could examine salient elements of these genres (e.g. AI in a strategy game, narrative in an adventure game and level design in a platformer) and attempt to determine the most effective methods of development through which to create them.

By learning from the findings and insight gained through the practical application of different development methodologies in this study, independent developers can gain a greater understanding of the importance of considering the development process of a game. Although both games developed for this project were not different enough for a direct comparative statistical study to be undertaken (quantitative playtesting feedback was generally too similar for both), they have demonstrated that the choice of a development model can and will affect the design, development process and player experience of a game in a number of ways. Key game areas that have been shown to be influenced by the choice of a predictive or adaptive development approach include narrative, challenge, visual presentation, difficulty distribution and element variety. Through these findings it has been shown that independent developers need to carefully consider the way in which they develop every aspect of their game, taking into account their own unique circumstances and game concept. With the low profit-margin and high failure rate in today's video game industry, neglecting to consider a development approach can not only result in an unprofitable and poorly received end result, but can even doom an entire game project to failure and cancellation before it is ever released.

References

Adams, E. (2010). *Fundamentals of Game Design* (Second ed.). Berkeley: New Riders.

Adler, P. S. (2005). The Evolving Object of Software Development. *Organization, 12*(3).

archetype. (n.d.). *Dictionary.com Unabridged*. Retrieved October 31, 2011, from http://dictionary.reference.com/browse/archetype

Baumgartner, E., Bell, P., Hoadley, C., Hsi, S., Joseph, D., Orrill, C., . . . Tabak, I. (2002). Design-Based Research: An Emerging Paradigm for Educational Inquiry. *Educational Researcher, 32*(1), 5-8.

Bohem, B. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes, 11*(4), 15-24.

Consalvo, M., & Dutton, N. (2006). Game Analysis: Developing a Methodological Toolkit for the Qualitative Study of Games. *Game Studies, 6*(1). Retrieved from http://gamestudies.org/0601/articles/consalvo_dutton

Davis, J., Steury, K., & Pagulayan, R. (2005). A Survey Method for Assessing Perceptions of a Game: The Consumer Playtest in Game Design. *Game Studies, 5*(1). Retrieved from http://www.gamestudies.org/0501/davis_steury_pagulayan/

Department of Health & Human Services - USA. (2008). *Selecting a Development Approach*. Retrieved 10th of March, 2011, from https://www.cms.gov/SystemLifeCycleFramework/downloads/SelectingDevelopmentApproach.pdf

Garratt, P. W. (1995). The Software Management Game. *Simulation & Gaming, 26*(1).

Grossman, A. (2003). *Postmortems from Game Developer*. USA: CMP Books.

Hayes, E., & Games, I. A. (2008). Making Computer Games and Design Thinking: A Review of Current Software and Strategies. *Games and Culture, 3*(3-4).

Huizinga, J. (1955/2005). Nature and Significance of Play as a Cultural Phenomenon. In K. Salen & E. Zimmerman (Eds.), *The Game Design Reader*. Cambridge: The MIT Press.

Ip, B. (2008). Technological, Content, and Market Convergence in the Games Industry. *Games and Culture, 3*(2).

IpsosMediaCT. (2010). 2010 Essential Facts About the Computer and Video Game Industry. USA: The Entertainment Software Association.

Irwin, M. J. (2008). Cooking Up a Blockbuster Game. Retrieved 14th of April, 2011, from http://www.forbes.com/2008/11/21/games-eedar-developers-tech-ebiz-cx_mji_1121eedar.html?partner=yahootix

Jarvinen, A. (2008). *Games without Frontiers: Theories and Methods for Game Studies and Design.* Ph.D., University of Tampere, Tampere.

Juul, J. (2005). Video Games and the Classic Game Model & Rules *Half Real: Video Games Between Real Rules and Fictional Worlds* (pp. 23-67). Massachusetts: The MIT Press.

Keith, C. (2010). *Agile Game Development with Scrum*. Crawfordsville: Addision Wesley Professional.

Kerr, A. (2006). Digital Games As Cultural Industry. *The Business and Culture of Digital Games*. Gateshead: Sage Publications.

Martin, A. (2000). The Design and Evolution of a Simulation/Game for Teaching Information Systems Development. *Simulation & Gaming, 31*(4).

Nutt, C. (2011). GDC 2011: Indie Revelations From Experienced Developers. Retrieved 11th of March, 2011, from http://gamasutra.com/view/news/33288/GDC_2011_Indie_Revelations_From_Experienced_Developers.php

O'Brien, R. (1998). An Overview of the Methodological Approach of Action Research Retrieved 12th of March, 2011, from http://www.web.net/~robrien/papers/arfinal.html

Oxland, K. (2004). Introduction to Game Design *Gameplay and Design* (pp. 7-23). Essex: Pearson Education.

Rollings, A., & Adams, E. (2003). The Internal Economy of Games and Game Balancing *On Game Design* (pp. 239-285). USA: New Riders Publishing.

Royce, W. (1970). *Managing the Development of Large Software Systems.* Paper presented at the IEEE WESCON 26.

Salen, K., & Zimmerman, E. (2004). Defining Games *Rules of Play: Game Design Fundamentals*. Cambridge: MIT Press.

Shakar, V. & Bayus, B. (2002). Network effects and competition: An empirical analysis of the home video game industry. *Strategic Management Journal, 24*(4).

Stevens, J. (2009). Practical Tips for Independent Game Development. Retrieved 12th of March, 2011, from http://www.gamedev.net/page/resources/_//feature/fgame-industry/practical-tips-for-independent-game-development-r2687

Appendices

Appendix 1 – Predictive game design document

Concept/Premise

The player is a tribal 'healer' on a secluded island, able to restore life to one thing by taking it from another. One day a powerful curse descends on the inhabitants of the island, causing them to turn into mindless undead creatures. As the only inhabitant immune to the curse, the healer tasks themselves with finding each person on the island and bringing them back to life.

Prologue

The game opens with a series of text explanations introducing the player to the setting, overlayed over a zoomed in scroll through of a number of levels (to show the cursed). The explanations will read:

- "You are a healer, the only person on your tribe's small island who holds natural power over the flow of life and death"
- "In violation of nature, your tribe attempted to gain dominion over life and death without trade, and thus were cursed"
- "To save your people, the ancient law of healing must be obeyed: Life that is to be given must first be taken…"

Epilogue

The final level of the game will begin with a short cutscene of the saved tribespeople running into a safe area, being pursued by a cursed. One will trip and be killed by the cursed, and the others will reach the safe area and close the gate. The Player then enters the level and must first heal the cursed with a nearby tree, so they are not killed. They then have two options; enter the safe area and end the game but without saving every tribesperson (one was killed), or use their own health (there are no available plants/animals) to heal the dead tribesperson, dying in the process but potentially saving every tribesperson (depending on if they saved everyone in each previous level).

Mechanics

Healing

The player is able to heal the cursed in two ways. The main way involves draining the life from other living objects (plants or animals) and transferring it to any nearby undead. The other way involves the player using a portion of their own life and small life forms (insects, small plants) to restore life to the cursed incrementally. During and shortly after any amount of healing, a cursed becomes temporarily stunned, remaining stationary and harmless. When the player uses any healing ability they become stationary for the duration, making them vulnerable to other cursed.

AI

The cursed will roam the screen in a simple back-forth pattern, walking in a straight line until they hit a collider or platform edge and then reversing direction. Once a cursed is healed they will begin to run and no longer stop at platform edges, until they reach the level's exit. If the player or a healed tribesperson goes in front of a cursed, the cursed will let out a scream, become enraged and start to run towards the player/tribesperson. Enraged cursed run off edges and move faster than usual. If the cursed makes contact with the player or a healed person they will attack and kill their target.

Level Design

Each level of the game will be contained within a single screen (no camera scrolling) with a set number of cursed roaming the screen. Each level will have a preset number of cursed the player must heal before they are allowed to proceed, but it will be possible to heal every cursed in each level with some careful planning and thinking. Along with live objects used for healing (plants, animals) and cursed, levels will contain platforms, climbable ladders/ledges and gates with corresponding levers/pressure plates to open/close (one lever or pressure plate may open/close multiple gates).

Player Controls

The player has three main abilities (tied to three buttons) aside from movement. The first emits a field draining life from nearby plants/animals and transferring to any cursed in front of the player within range. The second drains the player's life and transfers it to the nearest cursed. The third pulls levers in order to open/close gates. The player can move left and right along the level, at either a walk or a run, and is able to jump based on their movement speed (stationary jumps up, walking jumps a small distance, running jumps a large distance).

Additionally, if a player is standing in front of a ladder they can climb it by pushing up/down, and they will automatically grab onto ledges if they jump close enough to them.

Challenge

In order to heal every cursed on each level, while avoiding getting themselves or other healed people killed, the player must deduce which people to heal first, which levers to pull (and when), which pressure plates to stand on, and which live objects to used on which cursed. In this sense, the game consists of a number of puzzle challenges, but contains a few traditional psychomotor platformer challenges through the timing of healing cursed, and in the running, jumping and climbing needed to escape when being chased.

Look & Feel

The two main visual themes of the game can be described as ‘tribal’ and ‘island’. ‘Tribal’ refers mainly to the game’s characters and the interactive level geometry. Levers, ladders and platforms will be built of bamboo in a fashion that would be possible without the use of any modern or industrial tools (simple, functional designs). The tribespeople will be dark skinned, with the player (as a ‘shamanistic’ healer character) wearing a wooden mask, leaf headdress and various ornamental decorations. The cursed will appear as shadow emanating a black aura in the shape of a man, which will dissipate when they are healed. The ‘island’ visual indicator refers more to the background and environment; the ocean will be visible and the landscape will be rocky and sandy. Additionally, the plants that the player will drain life from will include palm trees, hanging vines, mushrooms and flowers, designed to appear native to an environment such as a tropical island.

The audio of the game will be acoustic and natural sounding to fit into the tribal motif, but will be somewhat dark and brooding in order to convey the weight of the curse that has settled on the island. Overall, the audio should not appear computer generated (unlike that of the previous game).

The player will be able to control their heal character with a moderate amount of agility; they will be able to turn quickly and move in ways that may not necessarily be physically realistic, but will make the game easier to play. The player and healed tribespeople will run at the same speed, while the cursed will run slightly faster, making them appear dangerous and unnatural.

Cursed will attack any non-cursed person they come into contact with, swinging their arms wildly in order to appear savage and out of control.

Asset Catalog

Character Models

- Player – Masked shaman healer
    - Idle
    - Run
    - Walk
    - Climb
    - Stationary jump
    - Moving jump
    - Fall
    - Ledge hang
    - Heal
    - Die
    - Sacrifice self (ending)
- Male tribesperson
    - Heal
    - Run
    - Fall
    - Die
- Female tribesperson
    - Heal
    - Run
    - Fall
    - Die
- Cursed
    - Walk
    - Chase
    - Attack
    - Fall
    - Healed

Scenery Models

- Platform – Bamboo
- Gate – Bamboo
- Lever – Bamboo
- Ladder – Rope
- Obstacle - Boulder
- Ground
- Rockwall
- Background/Ambient
    - Trees
    - Rocks
    - Bushes
    - Hanging rope
- Background skybox
- Pressure Plate

Heal Objects (With alive, dead variants)

- Palm Tree
- Vines
- Bush
- Mushrooms
- Flower
- Dragonfly
    - Fly animation

Sound Effects

- Player
    - Footsteps – Walk
    - Footsteps – Run
    - Footsteps - Land
    - Ledge grab
    - Heal – Using plant/animal

    - Heal – Using own health
    - Hit/Die
- Cursed
    - Enraged roar/scream
    - Footsteps – Walk
    - Footsteps – Run
    - Footsteps – Land
    - Attack
    - Healed
- Tribesperson
    - Hit/Die
    - Saved (Reach Exit)
    - (Re-use player footstep noises)
- Plant
    - Drain life (Decay)

Music

- Menu
- Level music 1
- Level music 2
- Ending short 1 (Save self)
- Ending short 2 (Sacrifice self)

Development Timeline

Week	Tasks
2	• Model and animate all player, cursed & tribesperson models • Script all player movement (linked with animations) and healing behaviours • Script enemy movement (linked with animations), healing behaviours and state switching (cursed – healed) • Model palm tree heal object (both states) and implement in game with full behaviours (including morphing between states).

	• Implement and test overall core healing mechanics.
3	• Model all heal objects with both states and implement healing behaviours • Model all scenery models (except background models) and implement pressure plate/lever relationship with gates (opening/closing) • Implement player interaction with scenery models (climbing ledges and ladders, pulling levers, stepping on pressure plates)
4	• Create background/ambient models • Texture all models • Construct pre-planned levels using fully scripted assets. • Create functional GUI (player health display, saved tribes people counter, pause)
5	• Create menu GUI & systems • Create prologue & epilogue scenes, scripting in cut scene functionality. • Implement audio • Finish any unfinished or delayed tasks • Polish, playtest & debug

Appendix 2 - Predictive level designs

Key:

Green – Heal object

Red – Cursed

Blue Arrow – Lever/Gate relationship

Double red arrow – Enraged cursed

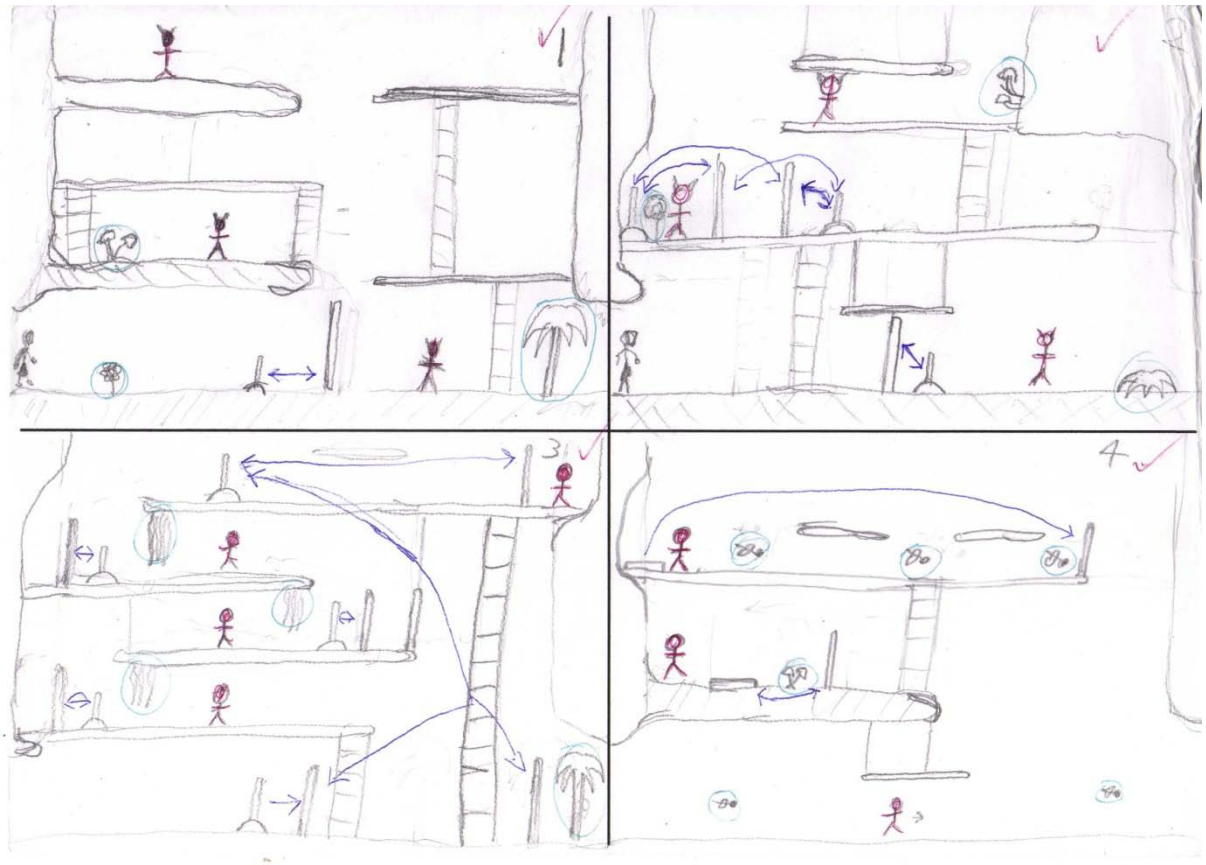

Figure 20. Cursed level designs 1 – 4.

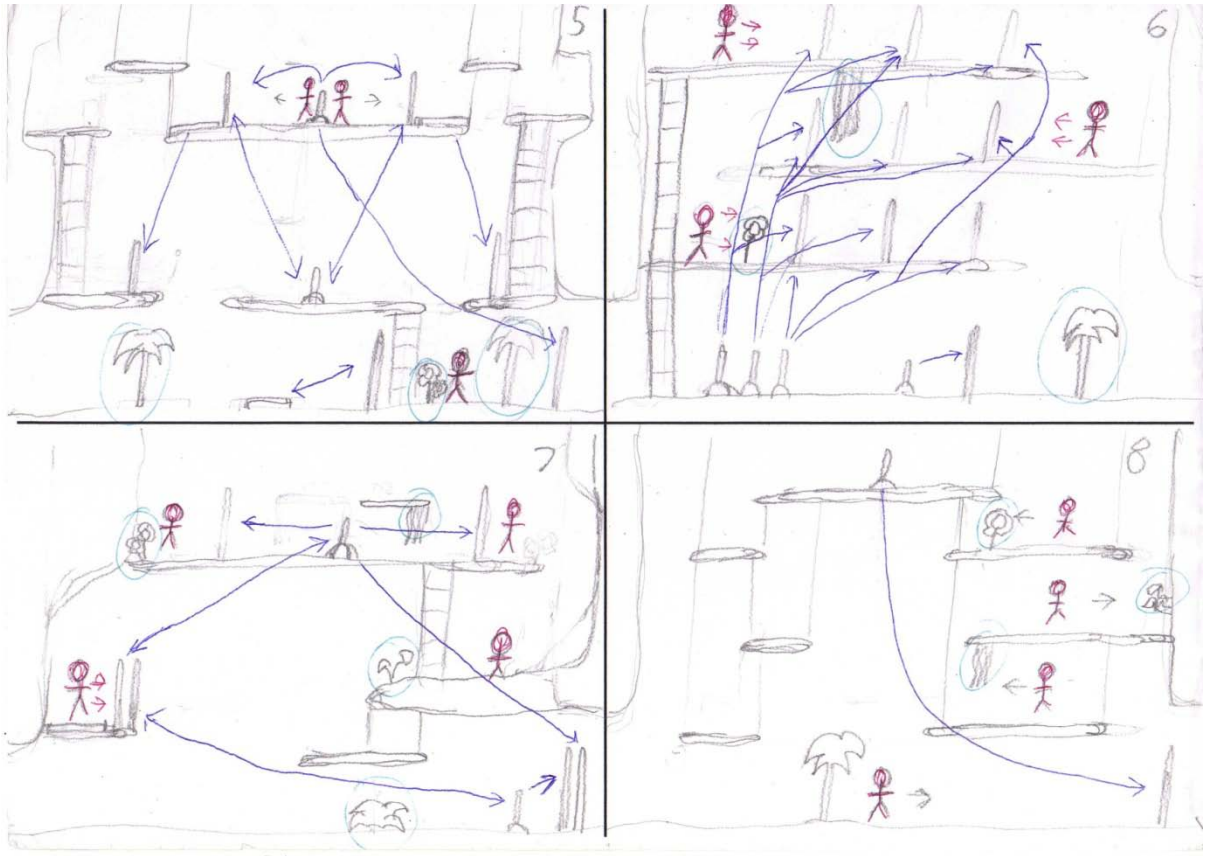

Figure 21. Cursed level designs 5 – 8.

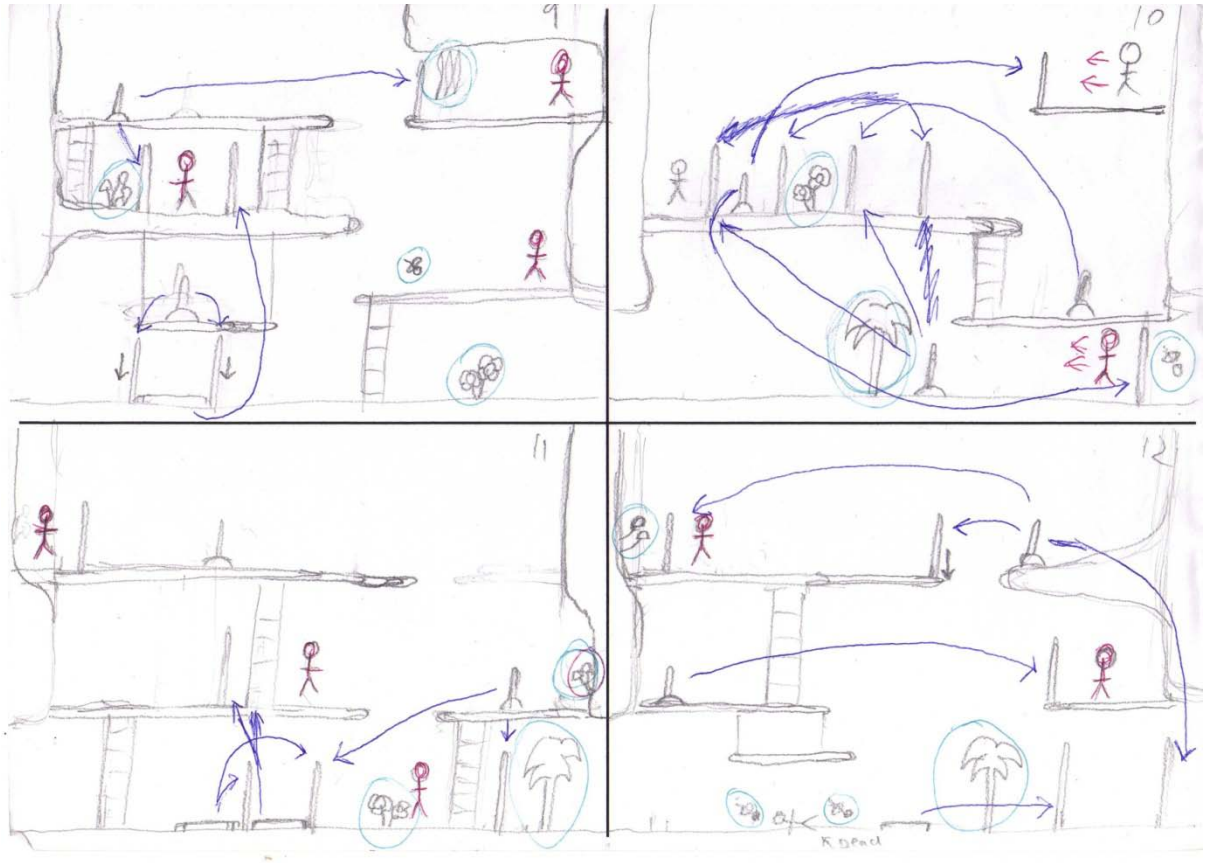

Figure 22. Cursed level designs 9 – 12.

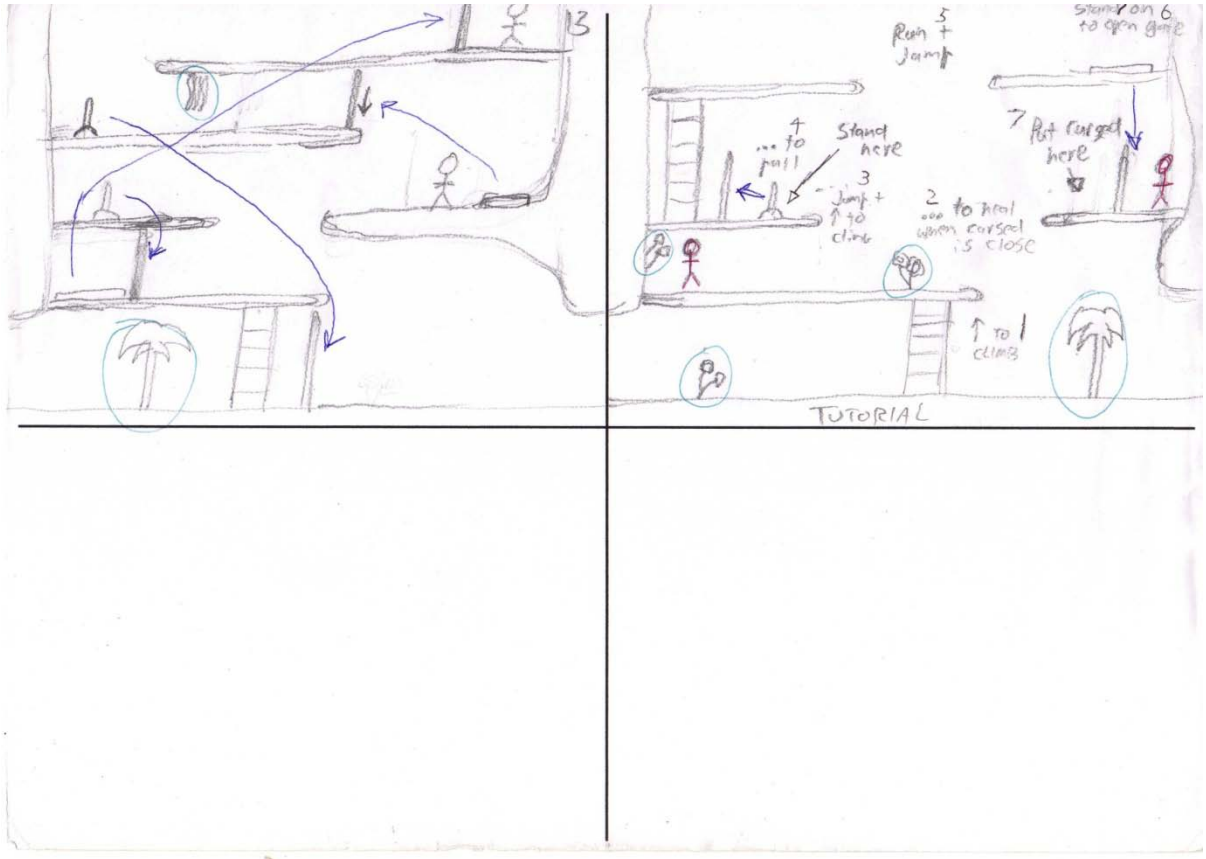

Figure 23. Cursed level designs 13 & Tutorial.

Appendix 3 - Sample predictive concept art

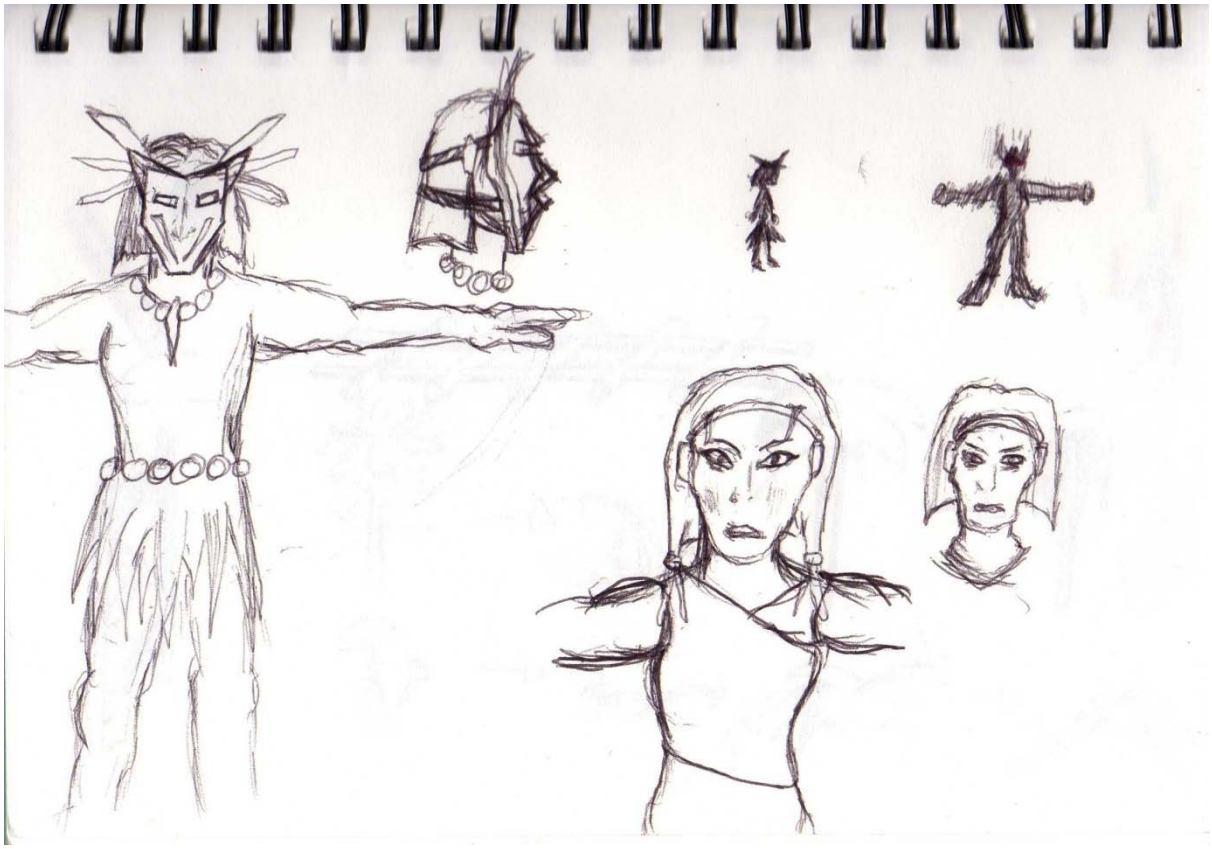

Figure 24. Cursed tribespeople concept art.

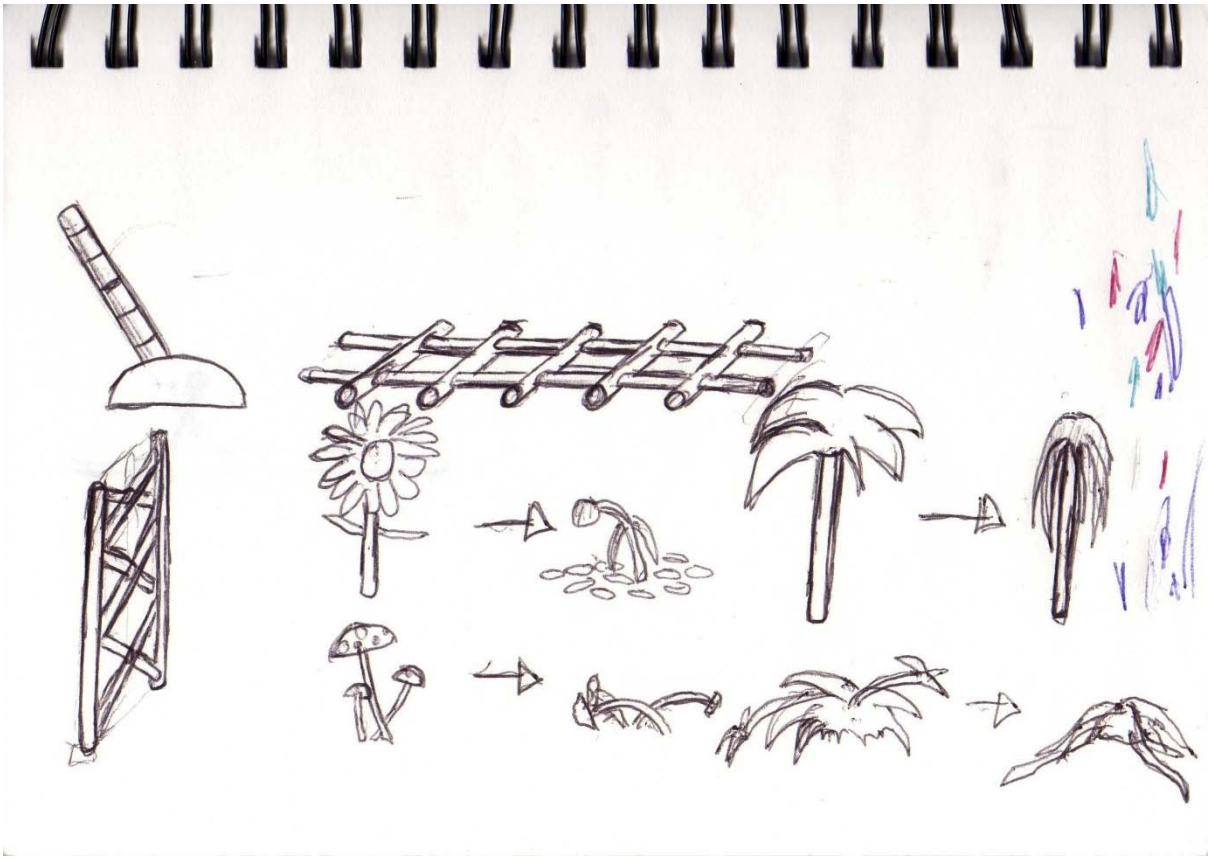

Figure 25. Cursed object concept art.

Appendix 4 – Final debugging change list

Dematerialized

- Added ‘Click to teleport’ prompt on prologue, if the player does not click within a certain time.
- Change the final outro text to ‘Thanks for participating’.
- Fixed a bug in the teleportation mechanic in order to improve to improve the accuracy of teleportation co-ordinate calculation.
- Enabled the up and down arrow keys.
- Added ‘Press Escape to skip’ dialogue in intro & outro.
- Added a picture in the controls showing the drop box and spiky cube as unkillable enemies.
- Widened some level areas to allow for more lenient teleportation.
- Removed a bug which could cause the final boss to damage the player on collision.

Cursed

- Added an extra collision barrier in order to stop a commonly reported bug of Healed becoming stuck on the right hand side level wall.
- Made the dragonfly heal object’s collision radius larger.
- Added a screen shake effect to make it more obvious when a Player is using their own health to heal something.
- Calculated the total number of saved tribespeople at the moment level changes, to avoid instances where they would not be added to the total if they escaped during the level fade out.
- Allowed right hand alt and shift keys to function as alternative controls for pulling levers and healing (respectively).
- Added a button in the pause menu that exits to the main menu.

Appendix 5 – Playtesting questionnaire

Please specify your level of agreement with the following statements:

The game was too difficult.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

The game did not provide enough of a challenge.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

There were areas in the game where I became frustrated.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

There were areas in the game where I was confused on how to progress.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

There game's instructions and guidelines were not made clear.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

The game was too short.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

There were noticeable bugs and technical errors.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

The controls and interface were difficult to use.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

I would want to play this game again.

Strongly Agree — Agree — Neutral — Disagree — Strongly Disagree

Please provide a score for the following game aspects (10 being the highest quality)

Presentation (Art Design/Aesthetics)

1 — 2 — 3 — 4 — 5 — 6 — 7 — 8 — 9 — 10

Music/Sound Effects

1 — 2 — 3 — 4 — 5 — 6 — 7 — 8 — 9 — 10

Graphics

1 — 2 — 3 — 4 — 5 — 6 — 7 — 8 — 9 — 10

Gameplay

1 — 2 — 3 — 4 — 5 — 6 — 7 — 8 — 9 — 10

Which parts of the game did you enjoy?

Which parts of the game did you dislike?

Are there any improvements or changes you could recommend?